

Maschinelles Lernen

Prof. Dr. Robert Hable

<https://www.statistik-hable.de>

Inhaltsverzeichnis

1	Grundlagen: Prognosen und Kausalität	5
2	Statistische Kennzahlen und Explorative Datenanalyse	12
2.1	Grundlagen: Arten von Daten	12
2.2	Untersuchung einzelner Variablen	13
2.2.1	Lagemaße, Streumaße und andere statistische Kennzahlen	13
2.2.2	Grundlegende grafische Darstellungsformen	20
2.3	Zusammenhang zwischen Variablen	24
2.3.1	Korrelationen	24
2.3.2	Grafische Darstellungsformen	26
3	Supervised Learning (I): parameterbasierte Verfahren	33
3.1	Einführung	33
3.2	Regression	35
3.2.1	Einfache lineare Regression	36
3.2.2	Multiple lineare Regression	36
3.3	(Binäre) Klassifikation: Logistische Regression	39
4	Supervised Learning (II): Nichtparametrische Verfahren	41
4.1	Einführung	41
4.2	Entscheidungsbäume	53
4.3	Neuronale Netze	62
4.4	k -Nearest Neighbors	68
5	Bewertung von Prognosen: Gütemaße im Supervised Learning	72
5.1	Einführung zu Gütemaße	72
5.2	Gütemaße in Regressionsproblemen	74
5.3	Gütemaße in Klassifikationsproblemen	78
5.4	Trainings- und Testdatensatz	81
5.5	Spezial: Schiefe Verteilung in der Zielvariable	83
6	Tuning Maschinelles Lernverfahren	88
6.1	Einleitung: Warum müssen Maschinelle Lernverfahren getuned werden?	88
6.2	Leave-One-Out	91

6.3	Kreuzvalidierung	92
7	Regularisierte Kern-basierte Verfahren (SVMs)	95
8	Universelle Konsistenz und Modellwahl	101
8.1	Universelle Konsistenz maschineller Lernverfahren	101
8.2	Modellwahl im maschinelles Lernen	104
8.2.1	Parametrische, semiparametrische und nichtparametrische Modelle	104
8.2.2	Beispiel: Additive Modelle mit L_2 -Boosting	108
9	Dimensionsreduktion	112
9.1	Einleitung: Der Fluch der hohen Dimension	112
9.2	LASSO	114
10	Zeitreihen und Online-Learning	117
10.1	Einleitung	117
10.2	Beispiel: Nachfragedaten	120
10.3	Trend und Saisonkomponenten	122
10.4	Zuwachs-Modell	126
10.5	Autoregressions-Modell	127
10.6	Testdatensatz bei Zeitreihen	128
10.7	Zusammenfassung und ergänzende Hinweise	130
11	Unsupervised Learning	132
11.1	Einführung	132
11.2	Clusteranalyse mit k -Means	135
11.3	Anomalieerkennung mit One-Class-SVM	136

1 Grundlagen: Prognosen und Kausalität

Deduktion und Induktion

Es gibt zwei Arten von Schlussfolgerungen: Deduktion und Induktion. Bei der Deduktion wird von einer allgemeingültigen Theorie mittels Logik auf Einzelfälle geschlossen. Bei der Induktion wird anhand von Einzelfällen eine allgemeine Theorie hergeleitet.

Beispiel für Deduktion:

- Jeder Mensch hat einen Kopf. Mein Nachbar ist ein Mensch. Also hat mein Nachbar einen Kopf.

Beispiele für Induktion:

- Mein Nachbar ist ein Mensch. Mein Nachbar hat einen Kopf. Also haben alle Menschen einen Kopf.
- Bei meinem Nachbarn haben die Globuli gewirkt. Also haben Globuli eine heilende Wirkung.

Während die Deduktion unproblematisch ist, ist die Induktion hochproblematisch. Aus einem oder wenigen Einzelfällen lässt sich nicht zuverlässig eine allgemein gültige Theorie herleiten. Anders sieht es aus, wenn man systematisch und viele Einzelfälle beobachtet. Dann lassen sich Theorien herleiten, die zumindest mit hoher Wahrscheinlichkeit stimmen:

Beispiele für Induktion basierend auf systematischen Beobachtungen:

- Geht morgen die Sonne wieder auf?

Stellen Sie sich vor, Sie sind der erste Mensch auf Erden und haben eben Ihren ersten Tag erlebt. Plötzlich stellen Sie erschrocken fest, dass die Sonne untergeht, und stellen sich die bange Frage, ob die Sonne morgen wieder aufgehen wird. Wie hoch ist die Wahrscheinlichkeit, dass die Sonne am 2. Tag wieder aufgehen wird?

Eine naheliegende Antwort ist $\frac{1}{2}$. Da man überhaupt nichts weiß und keine Vorerfahrung hat, mag man vielleicht von einer Fifty-Fifty-Chance ausgehen.

Stellen Sie sich weiter vor, dass die Sonne am 2. Tag tatsächlich wieder aufgeht. Wie hoch ist dann die Wahrscheinlichkeit, dass sie auch am 3. Tag wieder aufgeht?

Die Wahrscheinlichkeit ist sicherlich höher, denn Sie haben nun ja schon einmal erlebt, wie die Sonne wieder aufgegangen ist. Eine mögliche Antwort wäre $\frac{2}{3}$. Man kann sich das wie eine Urne vorstellen: Anfangs liegen eine weiße Kugel (= Sonne geht wieder auf) und eine schwarze Kugel (= ewige Finsternis) in der Urne. Nun haben Sie einmal erlebt, dass die Sonne wieder aufgegangen ist. Deshalb legen Sie eine zusätzliche weiße Kugel in die Urne. Nun befinden sich zwei weiße Kugeln und eine schwarze Kugel in der Urne, die Wahrscheinlichkeit für "weiß" (= Sonne geht wieder auf) beträgt nun also $\frac{2}{3}$.

Wenn die Sonne bisher an jedem Tag wieder aufgegangen ist, dann ist die Wahrscheinlichkeit, dass sie auch am nächsten Tag ($n + 1$) wieder aufgeht nach diesem Urnenmodell

$$\frac{n}{n + 1}$$

Man sieht hier an diesem zugegebenermaßen etwas sehr theoretischen Beispiel ein Grundprinzip sehr schön: Je mehr Tage vergehen, desto größer ist das n und desto mehr nähert sich die Wahrscheinlichkeit der Zahl 1 (=100%) an. Aber die Wahrscheinlichkeit bleibt immer kleiner als 1 – es gibt also immer ein Restrisiko.

Genau so ist es auch in der Wissenschaft: Je mehr Daten man sammelt, desto sicherer kann man sich sein, dass eine Theorie stimmt, aber hundertprozentige Sicherheit gibt es nie.

- Arzneimitteltests: Um die Wirksamkeit von neuen Medikamenten zu belegen, wird in der modernen Medizin ("Schulmedizin") heute nicht mehr auf Einzelerfahrungen von Einzelpersonen vertraut. Stattdessen werden nach strengen statistischen Regeln systematisch Versuche an vielen Personen durchgeführt ("Evidenzbasierte Medizin").

Statistik ist der Versuch durch mathematische Verfahren von systematisch durchgeführten Beobachtungen auf allgemeine Theorien zu schließen – und die Irrtumswahrscheinlichkeit so gering wie möglich zu halten. Irrtum ist aber nie ausgeschlossen!

Korrelation und Kausalität

Ein Zusammenhang (Korrelation), der sich in Daten findet, beweist noch keine Kausalität:

New evidence for the Theory of the Stork

Thomas Höfer^a, Hildegard Przyrembel^b and Silvia Verleger^c

^aFederal Institute for Risk Assessment, Berlin, ^bOffice of the National Breast Feeding Committee at BfR, Berlin, and ^cIndependent Midwife, Berlin, Germany

Summary

Correspondence:

Dr Thomas Höfer, Federal
Institute for Risk Assessment,
Thielallee 88-92, D-14195
Berlin, Germany.
E-mail:
thomas.hoefer@bfr.bund.de

Data from Berlin (Germany) show a significant correlation between the increase in the stork population around the city and the increase in deliveries outside city hospitals (out-of-hospital deliveries). However, there is no correlation between deliveries in hospital buildings (clinical deliveries) and the stork population. The decline in the number of pairs of storks in the German state of Lower Saxony between 1970 and 1985 correlated with the decrease of deliveries in that area. The nearly constant number of deliveries from 1985 to 1995 was associated with an unchanged stork population (no statistical significance). However, the relevance of the stork for the birth rate in that part of Germany remains unclear, because the number of out-of-hospital deliveries in this area is not well documented. A lack of statistical information on out-of-hospital deliveries in general is a severe handicap for further proof for the Theory of the Stork.

The intended value (disclaimer): This article is not intended to disprove the value of serious epidemiological investigations. It is an example of how studies based on popular belief and unsubstantiated theory, seconded by low quality references and supported by coincidental statistical association could lead to apparent scientific endorsement. Insofar it is a humorous case study for education in perinatal epidemiology.

Betrachten wir noch einmal den Fall von Globuli und Arzneimitteltests. Ein populärer Induktionsschluss ist:

Mein Nachbar hat Globuli genommen, danach ging es ihm besser.
Also wirken Globuli.

Der Schluss ist aus zwei Gründen problematisch:

1. Es ist ein Induktionsschluss basierend auf einer sehr kleinen Fallzahl.
2. Mit diesem "Versuchsdesign" ließe sich nur eine Korrelation feststellen und keine Kausalität! Andere mögliche Gründe für die gesundheitliche Besserung des Nachbarn:

- Die meisten gesundheitlichen Beschwerden heilt der Körper selber – auch ohne Medizin, Arzt oder Heilpraktiker. Dadurch bessern sich Beschwerden auch nach der Einnahme eines wirkungslosen Medikaments.
- Placebo-Effekt (Placebo-Effekt heißt nicht, dass man sich die gesundheitliche Besserung nur einbildet, der Placebo-Effekt hat tatsächlich eine messbare therapeutische Wirkung!)

Um Scheinkorrelationen auszuschließen und die kausale Wirkung des Medikaments (jenseits des Placebo-Effekts) so sicher wie möglich zu belegen, werden Arzneimitteltests in der evidenzbasierten Medizin unter äußerst strengen Versuchsdesigns durchgeführt: randomisierte doppelblinde Kontrollgruppen. Diese sehr wichtige Technik zum Nachweis von Kausalitäten behandeln wir später ausführlicher. Zunächst klären wir aber noch den Unterschied zwischen Statistik und Data Mining.

Data Mining und Statistik

Die **Statistik** ist eine strenge Wissenschaft zur Herleitung von allgemeinen Theorien aus möglichst systematisch beobachteten Einzelfällen (strenges Versuchsdesign, hoher Anspruch an die Datenqualität). Der Fokus liegt auf dem Verständnis von Zusammenhängen und der Herleitung von Kausalitäten.

Im **Data Mining** werden hauptsächlich auch statistische Methoden verwendet, aber der Fokus ist ein etwas anderer (wobei die Übergänge natürlich fließend sind und eine scharfe Abgrenzung nicht möglich ist). Im Data Mining liegt der Fokus auf dem Finden von “nützlichen” Mustern in den Daten, v.a. um gute Prognosen berechnen zu können. Für gute Prognosen können Scheinkorrelationen durchaus nützlich sein.

Beispiel:

Wenn die Schwalben tief fliegen, dann wird es bald ein Gewitter geben.

- Eine typische Data Mining Anwendung wäre nun, dieses Wissen zu benutzen, um Gewitter vorherzusagen. Obwohl es eine Scheinkorrelation ist und die tieffliegenden Schwalben das Gewitter nicht verursachen, ist der Zusammenhang nützlich für Prognosen.

- Eine typische statistische Fragestellung wäre die Überprüfung, ob tieffliegende Schwalben Gewitter verursachen. Dazu würden mehrere Gebiete über einen längeren Zeitraum untersucht, wobei man in einigen Gebieten zunächst die Schwalben entfernen würde und dann die Gewitterhäufigkeiten zwischen den Gebieten vergleichen würde.

Achtung: Häufig werden Prognosesysteme auch zur Steuerung verwendet. Das ist aber nur bei Kausalitäten und nicht bei Scheinkorrelationen sinnvoll. (Gewitter lassen sich eben nicht durch den Abschuss von Schwalben verhindern.)

Fallbeispiel aus einem meiner Unternehmensprojekte:

Ein Unternehmen produziert Stecker für den IT-Bereich. Die Produktionsdaten sollten analysiert werden mit dem Ziel, Muster zwischen den Produktionsbedingungen und fehlerhaften Steckern zu identifizieren. Tatsächlich wurde ein solches Muster gefunden: Fehler traten v.a. dann auf, wenn eine bestimmte Löttemperatur während des Produktionsvorgangs erhöht war. Es ist nun naheliegend zu folgern, dass die Fehler vermieden werden können, wenn die Löttemperatur besser eingestellt wird. Allerdings handelt es sich hierbei erstmal nur um eine Korrelation. Es ist nicht ausgeschlossen, dass es noch eine andere unbekannte Ursache gibt, die einerseits zu dem Fehler führt und andererseits aber auch die erhöhte Löttemperatur verursacht. In diesem Fall würde die Stabilisierung der Löttemperatur nicht automatisch die Fehler verhindern.

Hier ist tatsächlich Vorsicht geboten: Denn wenn das Unternehmen auf Ihren Rat hin viel Geld ausgibt, um die Löttemperatur zu stabilisieren, und dies nicht zum erwarteten Erfolg führt, dann kann das Ärger geben!

Nachweis von Kausalitäten

Von einer Beobachtungsstudie spricht man, wenn Daten ausgewertet werden, die nicht in einem kontrollierten Experiment entstanden sind. Beobachtungsstudien haben den Vorteil, dass nicht unbedingt extra Daten erhoben werden müssen, sondern häufig bereits vorhandene Daten verwendet werden können. Mit Beobachtungsstudien lassen sich in der Regel aber nur Korrelationen (Zusammenhänge) aufzeigen. Kausale Zusammenhänge können damit nicht belegt werden, hierzu werden kontrollierte Experimente benötigt. Ein Beispiel

einer Beobachtungsstudie haben wir schon bei der Untersuchung des Zusammenhangs zwischen der Anzahl an Störchen und der Anzahl an Geburten gesehen. Eine Beobachtungsstudie zum Thema Fleischkonsum vs. Vegetarismus wäre zum Beispiel, wenn man die Gesundheitsdaten von Vegetariern und Nicht-Vegetariern miteinander vergleichen würde. Im Gegensatz dazu berichtet uns die Bibel im Buch Daniel (Vers 1,8-16) von einem kontrollierten Experiment:

⁸ Daniel war entschlossen, sich nicht mit den Speisen und dem Wein der königlichen Tafel unrein zu machen, und er bat den Oberkämmerer darum, sich nicht unrein machen zu müssen. ⁹ Gott ließ ihn beim Oberkämmerer Wohlwollen und Nachsicht finden. ¹⁰ Der Oberkämmerer sagte aber zu Daniel: Ich fürchte mich vor meinem Herrn, dem König, der euch die Speisen und Getränke zugewiesen hat; er könnte finden, daß ihr schlechter ausseht als die anderen jungen Leute eures Alters; dann wäre durch eure Schuld mein Kopf beim König verwirkt. ¹¹ Da sagte Daniel zu dem Mann, den der Oberkämmerer als Aufseher für ihn selbst sowie für Hananja, Mischeel und Asarja eingesetzt hatte: ¹² Versuch es doch einmal zehn Tage lang mit deinen Knechten! Laß uns nur pflanzliche Nahrung zu essen und Wasser zu trinken geben! ¹³ Dann vergleiche unser Aussehen mit dem der jungen Leute, die von den Speisen des Königs essen. Je nachdem, was du dann siehst, verfahr weiter mit deinen Knechten! ¹⁴ Der Aufseher nahm ihren Vorschlag an und machte mit ihnen eine zehntägige Probe. ¹⁵ Am Ende der zehn Tage sahen sie besser und wohlgenährter aus als all die jungen Leute, die von den Speisen des Königs aßen. ¹⁶ Da ließ der Aufseher ihre Speisen und auch den Wein, den sie trinken sollten, beiseite und gab ihnen Pflanzkost.

Hierbei handelt es sich um ein kontrolliertes Experiment, das sehr viel besser für den Nachweis einer Kausalität

rein pflanzliche Ernährung $\longrightarrow$ besseres Aussehen

geeignet ist, als eine Beobachtungsstudie. Trotzdem ist das hier beschriebene Versuchsdesign aus heutiger wissenschaftlicher Sicht nicht sehr überzeugend. Dieses Versuchsdesign schließt nämlich einige möglicherweise verzerrende Effekte nicht aus.

Aufgabe: Überlegen Sie sich, welche verzerrenden Effekte es bei diesem Versuchsdesign geben könnte.

Verzerrende Effekte können (weitgehend) ausgeschlossen werden durch folgendes Versuchsdesign:

1. kontrolliert: eine Gruppe erhält die zu untersuchende Behandlung, eine separate Kontrollgruppe erhält die Behandlung nicht
2. randomisiert: die Einteilung in die beiden Gruppen erfolgt zufällig (niemand darf selbst entscheiden in welcher Gruppe er/sie ist)
3. doppel-blind: weder Versuchsperson noch Bewerter (z.B. Arzt) weiß, wer in welcher der beiden Gruppen ist (der Statistiker bei der Datenauswertung aber natürlich schon)

Dieses Versuchsdesign ist bei Medikamententests gesetzlich vorgeschrieben, wobei die Kontrollgruppe ein Placebo oder ein etabliertes Standardmedikament bekommt. Gesetzliche Krankenkassen bezahlen nur Medikamente, die bei solchen Tests signifikant besser abschneiden als das Placebo oder das bereits etablierte Standardmedikament. Homöopathische Mittel werden daher nicht¹ bezahlt, weil diese bei allen solchen Tests bisher durchgefallen sind und sich keine Wirkung (jenseits des Placeboeffekts) gezeigt hat.

¹Wenn Homöopathische Mittel bezahlt werden, dann ausschließlich als freiwillige Leistung zu reinen Marketingzwecken (wider besseren Wissens). Dies wird kritisiert, da die Krankenkassen dadurch aus rein wirtschaftlichen Interessen das Ansehen von Homöopathischen Mitteln noch weiter aufwerten.

2 Statistische Kennzahlen und Explorative Datenanalyse

2.1 Grundlagen: Arten von Daten

Wir müssen bei Daten verschiedene Arten oder Typen von Daten unterscheiden. Das ist keine akademische Spielerei, sondern von zentraler Bedeutung, weil die Wahl der zu verwendenden statistischen Verfahren vom jeweiligen Datentyp abhängt.

Unterscheidung des Datentyps nach dem Messskalenniveau

- kategoriell: für die Ausprägungen eines Merkmals sind nur vorgegebene Kategorien möglich; man muss unterscheiden zwischen
 - nominal: zwischen den Kategorien gibt es keine natürliche Rangfolge
Beispiele: Geschlecht, Blutgruppe, Religion, Geburtsort, ...
 - ordinal: zwischen den Kategorien gibt es eine natürliche Rangfolge
Beispiele: Schulnoten, Einkommensklassen, Ratingstufen von Ratingagenturen (AAA, AA+, AA, ...)
- kardinal / metrisch: (gemessene) Zahlenwerte, wobei die Abstände zwischen den Zahlenwerte eine sinnvolle inhaltliche Bedeutung haben (Bildung von Differenzen sinnvoll!)
Beispiele: Temperatur, Jahreszahl, Lebensalter, Volumen, Länge, ...

Bemerkungen und die häufigsten Fehlerquellen:

- alle Merkmalsausprägungen (insb. auch die nominalen Merkmale) lassen sich als Zahl kodieren; nur weil ein Merkmal aus Zahlen besteht, heißt das noch nicht, dass das Merkmal ordinal oder metrisch ist.
- ordinalskalierte Merkmale können durch Zahlen oder durch Buchstaben bzw. Wörter ausgedrückt werden
(Beispiel: Schulnoten im deutschen und im amerikanischen System)
- bei rein kategoriellen Merkmalen darf mit den Zahlen nicht gerechnet werden; insbesondere dürfen keine Durchschnitte gebildet werden
(Die Berechnung von Durchschnittsnoten durch Lehrer und Dozenten ist weitverbreitet aber eigentlich nicht korrekt.)

- die Einteilungen der unterschiedlichen Messskalien sind nicht ganz präzise; es gibt Überlappungen und Graubereiche

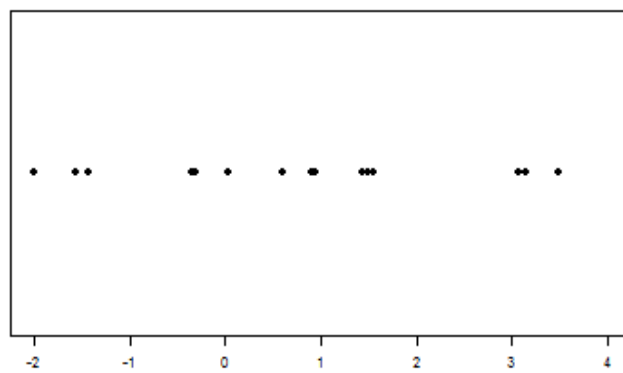
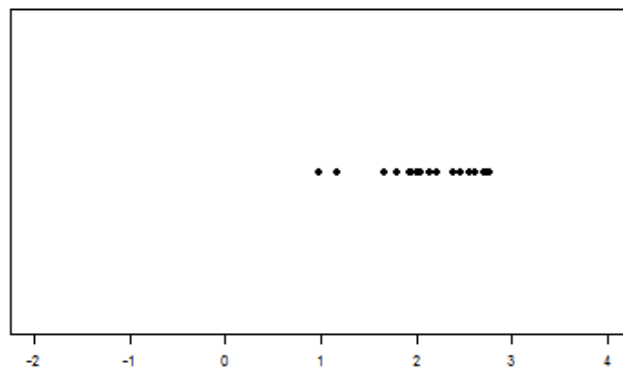
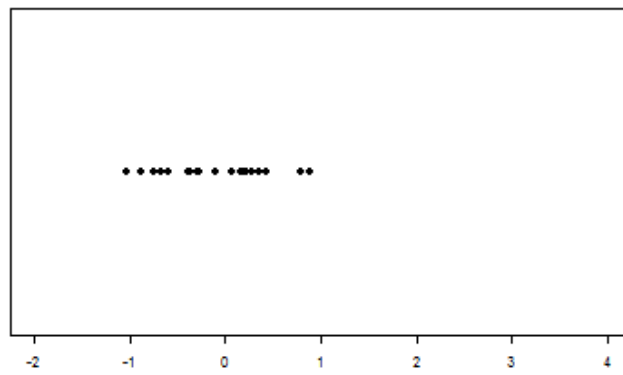
2.2 Untersuchung einzelner Variablen

Bevor man mit komplexeren statistischen Verfahren die Daten analysiert, sollte man sich erst einen umfassenden Überblick über die Daten verschaffen. Die nötigen Werkzeuge dazu werden in diesem Unterkapitel behandelt.

2.2.1 Lagemaße, Streumaße und andere statistische Kennzahlen

Die bekannteste statistische Kennzahl, um Daten zusammenzufassen, ist der Mittelwert. Der Mittelwert ist ein Lagemaß, weil er angibt, wo sich der “Schwerpunkt” der Daten befindet. Der Mittelwert ist nicht das einzige Lagemaß (und häufig auch nicht das am besten geeignete). Neben den Lagemaßen sind noch Streumaße von zentraler Bedeutung: diese geben an, wie stark die Daten um ihre Mitte streuen, also wie konzentriert sie sind.

Folgende 3 Bilder zeigen drei verschiedene Datensätze: Die Daten der ersten beiden Datensätze sind ähnlich stark um den jeweiligen Mittelpunkt konzentriert, unterscheiden sich aber deutlich in ihrer Lage; die Daten des dritten Datensatzes streuen wesentlich stärker.



Lagemaße gibt es für alle Skalenarten, aber nicht jedes Lagemaß ist bei jedem Skalenniveau sinnvoll. Streuungsmaße sind nur für metrisch skalierte Daten sinnvoll.

Hier eine kleine Übersicht, wo welche statistischen Kennzahlen sinnvoll sind:

Skalenniveau	Lagemaß	Streuungsmaß	Sonstige
nominal	Modus		Häufigkeiten (rel. und abs.)
ordinal	Median, Modus		Quantile, Häufigkeiten
metrisch	Mittelwert, Median, (Modus)	Standardabweichung, (Varianz), MAD Interquartilsabstand	Quantile, Max., Min.

Wir beginnen mit den Definitionen / der Berechnung der einzelnen Maße und wenden uns anschließend der wichtigsten Frage zu: Wann den Mittelwert und wann den Median benutzen?

Zunächst die Definitionen:

- Mittelwert: Das arithmetische Mittel aus reellen Zahlen $x_1, x_2, x_3, \dots, x_n$

$$\frac{x_1 + x_2 + x_3 + \dots + x_n}{n}.$$

- Median: der “mittlere” Wert

Der Median einer Datenreihe (reeller Zahlen) $x_1, x_2, x_3, \dots, x_n$ ist ein Wert m , so dass mindestens 50% der Daten kleiner oder gleich diesem Wert m sind und gleichzeitig mindestens 50% der Daten größer oder gleich diesem Wert m sind.

Grob gesprochen: Sortiere die Daten der Größe nach und nimm den Wert, der in der Mitte der Zahlenreihe steht.

Beispiel: Bilde den Median aus den Zahlen

76, 89, 73, 90, 61, 66, 82, 62, 79, 85, 86, 80, 86, 62.

Bilde den Median aus den Zahlen

1, 3, 2, 1, 0

Problem: Der Median ist nicht immer eindeutig, z.B. bei

2, 6, 5, 0

In diesem Fall nimmt man (meist) die Mitte zwischen dem kleinsten und dem größten möglichen Median.

- Modus: der häufigste Wert

Der Modus einer Datenreihe ist der Wert, der am häufigsten auftaucht. Er kann auch bei kategoriellen Werten verwendet werden und ist dann also die Kategorie, die am häufigsten auftaucht. (Bei metrischen Merkmalen spielt der Modus eine sehr untergeordnete Rolle.)

Beispiel: Folgende Datenreihe gibt die Augenfarbe für jede Gruppe einer Person wieder, was ist hier der Modus?

blau, blau, grün, braun, grün, grün, braun, grau

- Häufigkeiten: Vor allem bei nominalen Merkmalen sind Häufigkeitstabellen sehr üblich.

Beispiel: Titanic-Datensatz

Merkmal	Ausprägung	absolut	relativ (in %)
Klasse	1	325	14,8
	2	285	12,9
	3	706	32,1
	Crew	885	40,2
Geschlecht	m	1731	78,6
	w	470	21,4
Überlebt	Nein	1490	67,7
	Ja	711	32,3

- Quantile: Sei α eine Zahl zwischen 0 und 1.

Das α -Quantil einer Datenreihe (reeller Zahlen) $x_1, x_2, x_3, \dots, x_n$ ist ein Wert q , so dass mindestens $\alpha \cdot 100\%$ der Daten kleiner oder gleich diesem Wert q sind und gleichzeitig mindestens $(1 - \alpha) \cdot 100\%$ der Daten größer oder gleich diesem Wert q sind.

(Achtung: andere Konventionen zur Definition des Quantils sind möglich.)

Grob gesprochen ist das 25%-Quantil der Wert q so dass 25% der Daten kleiner als q sind (und entsprechend 75% der Daten größer als q sind).

Nehmen wir an, dass die Daten $x_1, x_2, x_3, \dots, x_n$ der Größe nach geordnet sind, wobei x_1 der kleinste und x_n der größte Wert ist. Das α -Quantil lässt sich dann folgendermaßen berechnen:

$$\alpha\text{-Quantil} = \begin{cases} \frac{1}{2}(x_{n \cdot \alpha} + x_{n \cdot \alpha + 1}) & n \cdot \alpha \text{ ganzzahlig} \\ x_{\lceil n \cdot \alpha \rceil} & \text{falls } n \cdot \alpha \text{ nicht ganzzahlig} \end{cases}$$

Hierbei bedeutet $\lceil n \cdot \alpha \rceil$, dass die Zahl $n \cdot \alpha$ aufgerundet wird.

Der Median ist ein Speziellfall eines Quantils, nämlich das 50%-Quantil.

Das 25%-Quantil und das 75%-Quantil benötigen wir später beim 5-Number-Summary und beim Boxplot.

- Maximum und Minimum: klar
- Standardabweichung: das wichtigste Streumaß

Um die Standardabweichung einer Datenreihe (reeller Zahlen) $x_1, x_2, x_3, \dots, x_n$ zu berechnen, berechnet man zunächst den Mittelwert $\bar{x}$ der Daten; dann ist die Standardabweichung gleich

$$\sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \dots + (x_n - \bar{x})^2}{n - 1}}.$$

Als grobe Fausregel gilt, dass vielfach (aber nicht immer) etwa 68% der Daten bis zur Länge von 1 Standardabweichung vom Mittelwert entfernt liegen und etwa 95% der Daten bis zur Länge von 2 Standardabweichungen vom Mittelwert entfernt liegen. Dies ist die sogenannte *68-95-Regel*.

- MAD: eine robuste Alternative zur Standardabweichung

Mittelwert und Standardabweichung sind empfindlich gegenüber Ausreißern. Das heißt, einzelne große Werte im Datensatz haben einen sehr starken Einfluss auf den Mittelwert und die Standardabweichung. Eine robuste (d.h. gegenüber Ausreißern unempfindliche) Alternative zum Mittelwert ist der Median. Eine robuste Alternative zur Standardabweichung ist der MAD ("Median Absolute Deviation").

Für eine Datenreihe (reeller Zahlen) $x_1, x_2, x_3, \dots, x_n$ bechnen wir hierzu zunächst den Median m . Der MAD ist dann

der Median der Werte $|x_1 - m|, |x_2 - m|, |x_3 - m|, \dots, |x_n - m|$

- Interquartilsabstand: weitere robuste Alternative zur Standardabweichung

Der Interquartilsabstand ist eine weitere robuste Alternative zur Standardabweichung und wird später beim Boxplot benötigt.

Der Interquartilsabstand ist die Differenz zwischen dem 75%-Quantil und dem 25%-Quantil.

Für einen ersten Überblick über die Daten, sollte man zunächst für jede Variable einzeln eine Auswahl sinnvoller statistischer Kennzahlen berechnen und analysieren.

Üblich sind beispielsweise 5-Number-Summeries der Form

Anzahl der Daten und Maßeinheit	
Median	
25%-Quantil	75%-Quantil
Minimum	Maximum

Wann den Mittelwert und wann den Median verwenden?

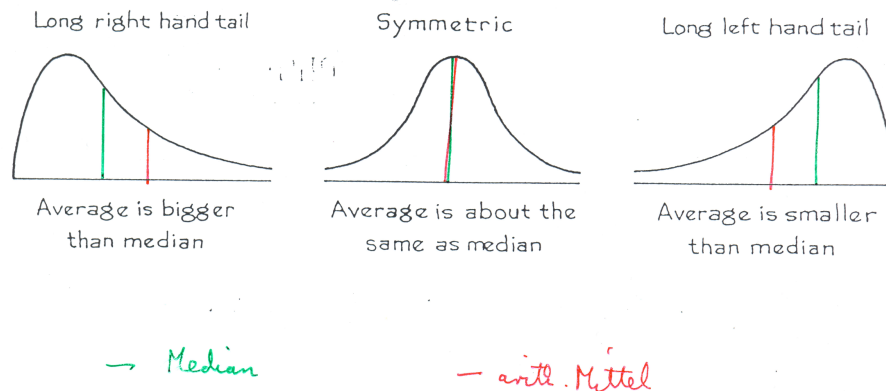
Bei ordinal skalierten Daten muss der Median verwendet werden, weil der Mittelwert nicht anwendbar ist. Bei metrischen Daten hat man die Wahl: Mittelwert oder Median?

Bei metrischen Daten ist der Mittelwert der Standard, aber es gibt Umstände, unter denen der Median zu bevorzugen ist:

- Ausreiser: Wie oben erwähnt, ist der Mittelwert empfindlich gegenüber Ausreisern. Das heißt, einzelne große Werte im Datensatz haben einen sehr starken Einfluss auf den Mittelwert. Der Median ist hingegen unempfindlich gegenüber Ausreisern ("robust"). Bei Vorhandensein von Ausreisern kann es daher ratsam sein, den Median statt den Mittelwert zu verwenden.

- Schiefe Verteilung:

Wenn die Dichtefunktion einer Wahrscheinlichkeitsverteilung symmetrisch ist, dann sind Mittelwert und Median identisch. Bei unsymmetrischen Dichtefunktionen können Mittelwert und Median erheblich voneinander abweichen:



Rechtsschiefe Verteilung: die Dichtefunktion ist am rechten Ende weiter “rausgezogen” (der Buckel ist dann links von der Mitte).

Linksschiefe Verteilung: die Dichtefunktion ist am linken Ende weiter “rausgezogen” (der Buckel ist dann rechts von der Mitte).

Bei schiefen Verteilungen ist der Mittelwert als Lageparameter meist weniger aussagekräftig, weil er zu sehr zum Rand hin ausgelenkt wird; der Median beschreibt das Zentrum der Daten besser. Bei schiefen Verteilungen treten häufig auch Ausreiser auf.

Ob die Daten aus einer schiefen Verteilung stammen, sieht man grafisch am Boxplot oder im Histogramm (vgl. später).

- Inhaltliche Gründe:

Typische Fragestellungen, bei denen der Median inhaltlich sinnvoller ist als der Mittelwert sind z.B. Armut und Einkommen.

Ob man arm oder reich ist, hängt weniger davon ab, ob man mehr oder weniger als das Durchschnittsvermögen besitzt, sondern ob man zur reicheren Bevölkerungshälfte oder zur ärmeren Bevölkerungshälfte

zählt – ob man also mehr oder weniger als den Median der Vermögen besitzt.

Entsprechend gilt beim Einkommen in Deutschland als armutsgefährdet, wer weniger als 60% des Medianeinkommens hat.

Wenn Sie in einer Branche oder einem Unternehmen von hohen Durchschnittsgehältern hören, dann hat das erstmal noch nicht so viel zu sagen. (Dem normalen Mitarbeiter nützt es nichts, dass das astronomische Gehalt des Geschäftsführers das Durchschnittsgehalt nach oben auslenkt. Ein besseres Bild über die wahren Gehälter gibt Ihnen der Median.)

Wenn inhaltliche Gründe für den Median sprechen, dann ist die Verteilung meistens auch schief (und es gibt häufig Ausreiser).

Tipp: In der Praxis ist es ein gutes Vorgehen, sowohl den Mittelwert als auch den Median auszurechnen. Wenn beide Werte in etwa übereinstimmen, dann ist es wohl nicht so wichtig, was man nimmt. Falls sich beide Werte deutlich voneinander unterscheiden, dann sollte man intensiv nachdenken, woran es liegt und was besser ist.

2.2.2 Grundlegende grafische Darstellungsformen

Die grafischen Darstellungsformen hängen wieder von der Skalenart der Daten ab. Wir beginnen wieder mit einer kleinen Übersicht, wo welche Darstellungsart sinnvoll sind:

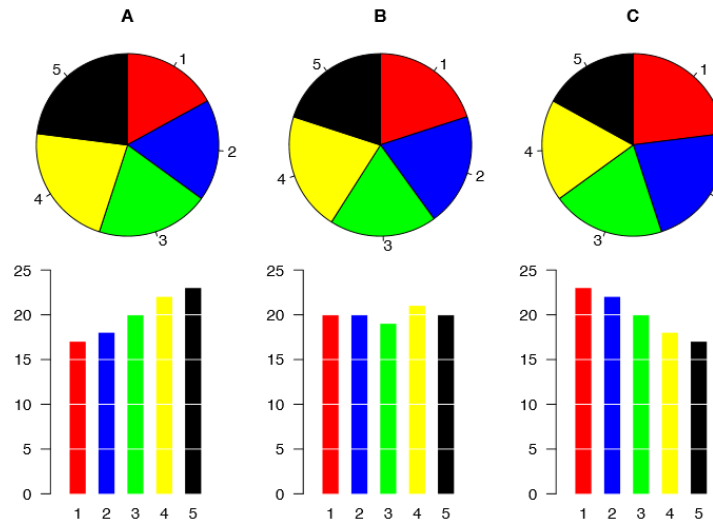
Sklenniveau	Darstellungsarten
nominal	Balkendiagramm, (Tortendiagramm)
ordinal	Boxplot, Balkendiagramm
metrisch	Boxplot, Histogramm

Nun die einzelnen Beschreibungen; die Umsetzung in R behandeln wir im weiteren Verlauf der Vorlesung.

- Tortendiagramm:

Tortendiagramme sind – v.a. in den Medien – äußerst verbreitet, bei Statistikerinnen und Statistikern sind Tortendiagramme allerdings äußerst unbeliebt. Grund hierfür ist, dass das menschliche Auge die Größe von

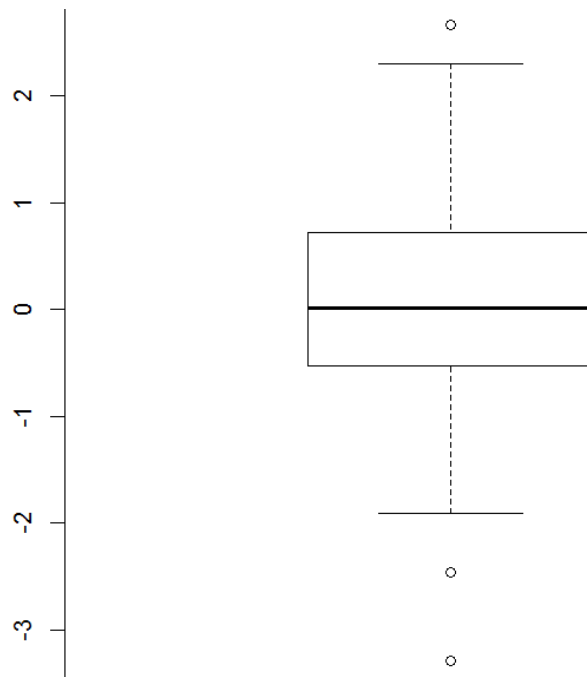
Flächen und Winkel nur schwer vergleichen kann. Viel besser geeignet sind stattdessen Balkendiagramme. Folgendes Beispiel zeigt, warum man Tortendiagramme nicht verwenden sollte. Welche Fläche ist die größte?



aus dem englischen Wikipedia-Artikel zu "pie chart"

<https://commons.wikimedia.org/wiki/File:Piecharts.svg>

- Balkendiagramm: kennt jeder; siehe obiges Beispiel zum Tortendiagramm
- Boxplot: kompakte grafische Darstellung der 5-Number-Summary
Beispiel eines Boxplots:



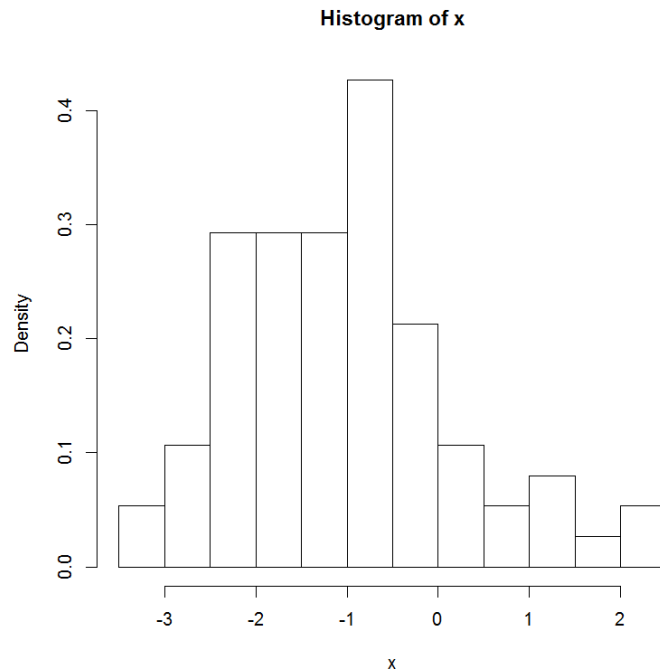
Die Höhe der Box entspricht genau dem Interquartilsabstand und illustriert somit die Streuung der Daten.

Whisker: Für die Länge der Whisker wird zunächst der Interquartilsabstand mit dem Wert 1,5 multipliziert. Dann werden die Whisker bis zum ersten Datenpunkt nach innen verschoben.

Alle Daten die außerhalb der Whisker liegen, werden einzeln als Punkte eingezeichnet und gelten als "Ausreißer".

- Histogramm: anschauliche Darstellung der Verteilung der Daten (entspricht der Dichtefunktion von Wahrscheinlichkeitsverteilungen)

Hier ein Beispiel für ein Histogramm:



Obiges Histogramm sieht fast wie ein Balkendiagramm aus, aber es gibt einen wichtigen Unterschied: Der Anteil der Daten, die in ein Intervall (auf der x -Achse) fällt, wird nicht durch die Balkenhöhe angegeben, sondern durch die Fläche der Balken. Obiges Histogramm wurde aus 75 Daten gebildet. Wieviele Datenpunkte jeweils in welches Intervall (auf der x -Achse) fallen, ist in folgender Tabelle angegeben

Intervall	1	2	3	4	5	6	7	8	9	10	11	12
Anzahl Datenpunkte	2	4	11	11	11	16	8	4	2	3	1	2

Allgemeiner Aufbau eines Histogramms:

- Das Histogramm stellt Prozentzahlen als Fläche dar.
- Die vertikale Skala ist die “Dichteskala”: Prozentzahl der Fälle pro Einheit der horizontalen Skala
- Die Höhe eines Balken/Blockes ergibt sich durch Division: Höhe $\times$ Grundseite = Prozentsatz der Daten in dem jeweiligen Intervall
- Die Gesamtfläche ist $1 = 100\%$.

2.3 Zusammenhang zwischen Variablen

In diesem Abschnitt werden Werkzeuge behandelt, mit denen man einfache Zusammenhänge zwischen zwei Variablen untersuchen kann: zum einen durch Kennzahlen (Korrelationen), zum anderen durch grafische Darstellungen.

2.3.1 Korrelationen

Korrelationen sind statistische Kennzahlen, mit denen Zusammenhänge zwischen zwei Variablen beschrieben werden können.

Eine Warnung vorneweg: Korrelation bedeutet nicht Kausalität! (Das klingt einleuchtend. Trotzdem ist es – vor allem in den Medien – fast schon die Regel, dass ein unabweisbar vorhandener Zusammenhang als Beleg für eine völlig spekulative Kausalität verwendet wird (v.a. bei Ernährungs- und Gesundheitsthemen).

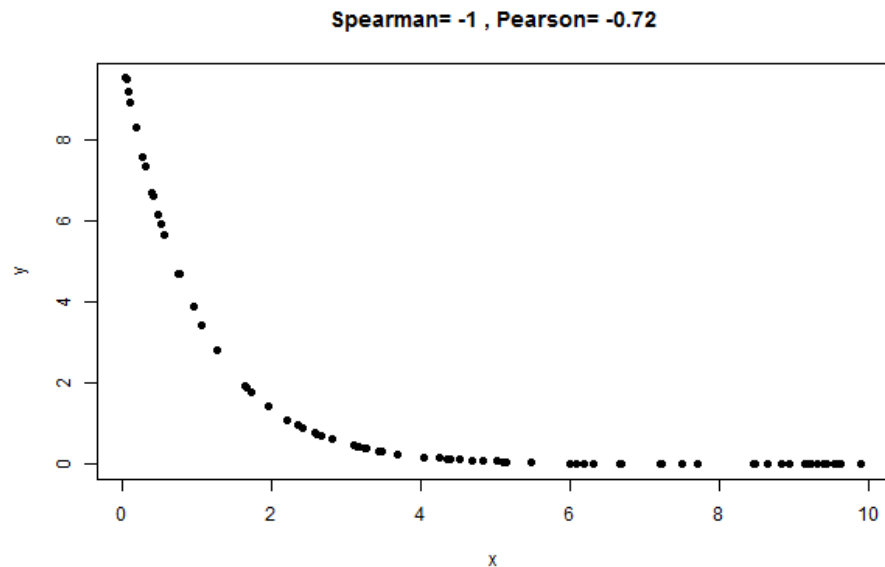
Es gibt verschiedene alternative Maßzahlen zur Korrelation. Wir behandeln zwei solcher Korrelationskoeffizienten.

- Pearson-Korrelationskoeffizient
 - der “gewöhnliche” Korrelationskoeffizient (spricht man von einer Korrelation ohne genauere Angaben, dann ist normalerweise dieser Korrelationskoeffizient gemeint)
 - nur verwendbar für metrisch skalierte Daten
 - nur geeignet für lineare Zusammenhänge (vgl. nachfolgende Bilder)
 - eine Zahl zwischen -1 und 1 mit folgender Interpretation:
 - * eine Zahl um die 0 bedeutet (fast) kein Zusammenhang
 - * je weiter die Zahl von der 0 weg ist, desto stärker ist der Zusammenhang
 - * positive Zahlen bedeuten einen positiven Zusammenhang (“je größer ...desto größer ...”)
 - * negative Zahlen bedeuten einen negativen Zusammenhang (“je größer ...desto kleiner ...”)
 - Berechnung: Ein Datensatz bestehe aus n Datenpunkten wobei $x_1, x_2, \dots, x_n$ die Werte eines metrischen Merkmals (eine Spalte

im Datensatz) und $y_1, y_2, \dots, y_n$ die Werte eines anderen metrischen Merkmals (eine andere Spalte im Datensatz) seien. Sei $\bar{x}$ der Mittelwert der x -Werte und $\bar{y}$ der Mittelwert der y -Werte. Dann ist der Pearson-Korrelationskoeffizient gleich

$$\frac{\sum_{i=1}^n (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \cdot \sum_{i=1}^n (y_i - \bar{y})^2}}$$

- Spearman-Korrelationskoeffizient
 - eine Alternative zum Pearson-Korrelationskoeffizient, die breiter einsetzbar ist (und unempfindlich gegenüber Ausreißern ist)
 - verwendbar für ordinal und metrisch skalierte Daten
 - nicht nur geeignet für lineare Zusammenhänge, sondern für alle monotonen Zusammenhänge (aber nicht geeignet für U-förmige Zusammenhänge)
 - eine Zahl zwischen -1 und 1 mit folgender Interpretation:
 - * eine Zahl um die 0 bedeutet (fast) kein Zusammenhang
 - * je weiter die Zahl von der 0 weg ist, desto stärker ist der Zusammenhang
 - * positive Zahlen bedeuten einen positiven Zusammenhang (“je größer ... desto größer ...”)
 - * negative Zahlen bedeuten einen negativen Zusammenhang (“je größer ... desto kleiner ...”)
 - Berechnung: Wie der Pearson-Korrelationskoeffizient, nur das anstelle der Datenwerte x_i und y_i die entsprechenden Ränge verwendet werden. Hierzu werden die Werte $x_1, x_2, \dots, x_n$ der Größe nach sortiert. Der Rang von x_i ist dann die “Platznummer” von x_i nach der Sortierung. Das heißt, der kleinste Wert hat den Rang 1, der zweitkleinste Wert den Rang 2, ... der größte Wert den Rang n . Entsprechend wird bei den Werten $y_1, y_2, \dots, y_n$ vorgegangen. Anschließend wird der Korrelationskoeffizient nach derselben Formel wie beim Pearson-Korrelationskoeffizient berechnet, nur das anstelle der Datenwerte x_i und y_i die entsprechenden Ränge eingesetzt werden.
 - Beispiel:



2.3.2 Grafische Darstellungsformen

Bei der grafischen Darstellung von Zusammenhängen kommt es auch wieder entscheidend auf die Skalenarten der Merkmale an, die miteinander in Beziehung gesetzt werden. Eine Übersicht über die jeweils geeigneten Darstellungsformen gibt folgende Tabelle:

		Variable 1	
		nominal (ordinal)	metrisch (ordinal)
Variable 2	nominal (ordinal)	Kontingenztafel, Balkendiagramm	Boxplots
	metrisch (ordinal)	Boxplots	Streudiagramm

Bei den ordinalen Daten können alle hier genannten grafischen Darstellungsformen verwendet werden – sowohl die für nominale als auch die für metrische

Daten. Je nachdem, wie viele verschiedene Kategorien es bei den ordinalen Daten gibt, sind die Darstellungsformen für nominale Daten (bei wenigen Kategorien) oder die Darstellungen für metrische Daten (bei vielen Kategorien) geeigneter.

- Zusammenhang zwischen zwei (oder mehreren) nominalskalierten Merkmalen

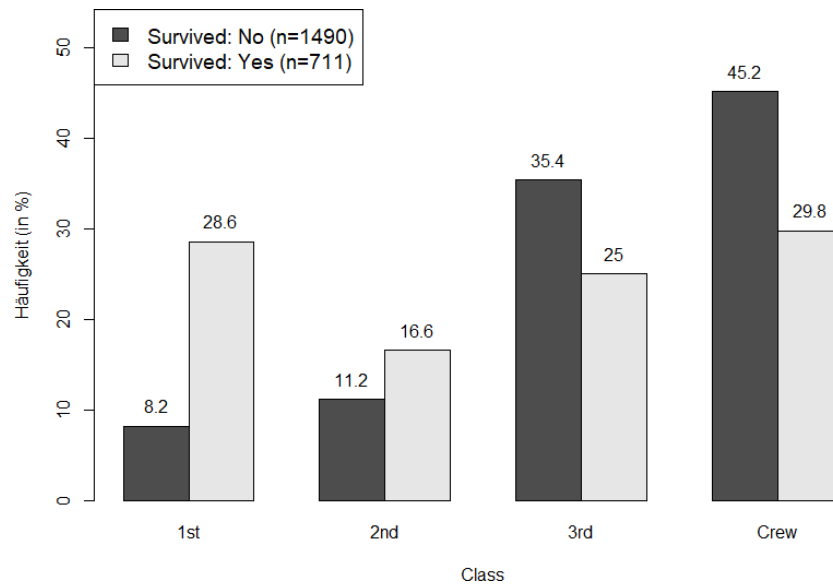
Zusammenhänge zwischen zwei nominalskalierten Merkmalen lassen sich gut mit Hilfe von Kontingenztafeln (auch: Kreuztabellen) darstellen. Dabei wird in einer Tabelle ein Merkmal (in den Zeilen der Tabelle) einem anderen Merkmal (in den Spalten) gegenübergestellt und die entsprechenden Häufigkeiten eingetragen.

Hier ein Beispiel für den Titanic-Datensatz:

Klasse \ Überlebt	Nein	Ja	Randhäufigkeit
1	122	203	325
2	167	118	285
3	528	178	706
Crew	673	212	885
Randhäufigkeit	1490	711	2201

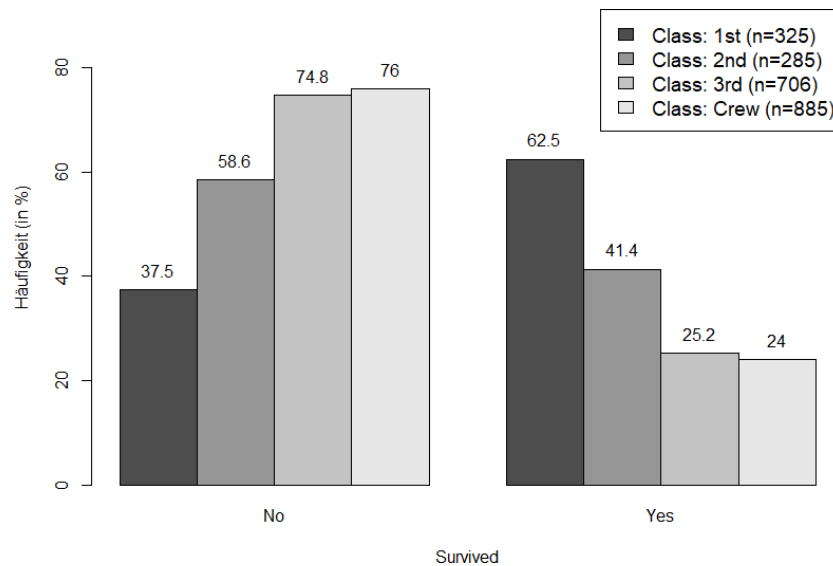
Ablesebeispiele: Bei den Passagieren der 2. Klasse haben 167 nicht überlebt und 118 haben überlebt. Insgesamt gab es 706 Passagiere der 3. Klasse. Unter allen Passagieren haben 1490 nicht überlebt.

Oftmals ist es zweckmäßiger anstelle der absoluten Häufigkeiten die relativen Häufigkeiten anzugeben. Diese lassen sich dann anschaulich als Balkendiagramm darstellen, wie in diesem Beispiel:



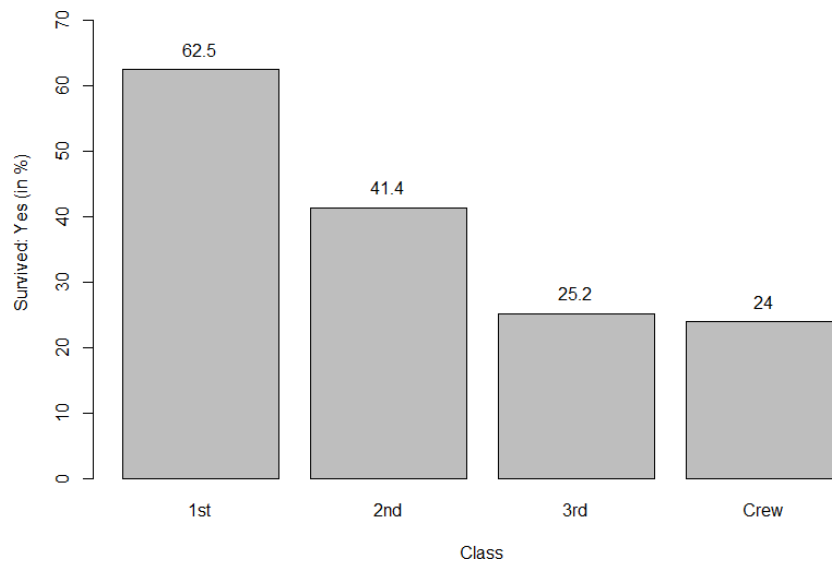
In diesem Balkendiagramm geben die Prozentzahlen der schwarzen Balken an, wie sich die verstorbenen Passagiere (No, $n = 1490$) auf die verschiedenen Klassen verteilen. Die Prozentzahlen der grauen Balken geben jeweils an, wie sich die überlebenden Passagiere (Yes, $n = 711$) auf die verschiedenen Klassen verteilen. Entsprechend addieren sich die schwarzen Balken auf 100 % und die grauen Balken addieren sich ebenfalls auf 100 %.

Diese Prozentberechnung ist jedoch nicht gut geeignet, um den Zusammenhang zwischen der Klasse und dem Überleben (No/Yes) darzustellen. Interessant ist ja, inwieweit das Überleben (Zielvariable) von der Klassenzugehörigkeit (Einflussvariable) abhängt. Dies ist wesentlich besser in folgendem Balkendiagramm dargestellt:



Hierbei ist nun für jede Klasse dargestellt, wie viel Prozent der Passagiere verstorben sind (linke Balkegruppe) und wieviel Prozent der Passagiere überlebt haben (rechte Balkengruppe). Nun sieht man sehr schön, dass Passagiere der 1. Klasse die höchste Überlebenswahrscheinlichkeit (62,5 %) hatten. Die Überlebenswahrscheinlichkeit in der 3. Klasse war mit 25,2 % in etwa so niedrig, wie die Überlebenswahrscheinlichkeit der Crew-Mitglieder (24 %).

Da die Zielvariable Survived nur aus zwei Kategorien (Yes/No) besteht, ist dieses Balkendiagramm jedoch unnötig aufgebläht. Die linke Balkengruppe ist einfach nur eine Spiegelung der rechten Balkengruppe. Das Balkendiagramm kann daher verschlankt werden, in dem nur die rechte Balkengruppe (Survived = Yes) dargestellt wird (damit kann dann auch auf die Legende verzichtet werden):

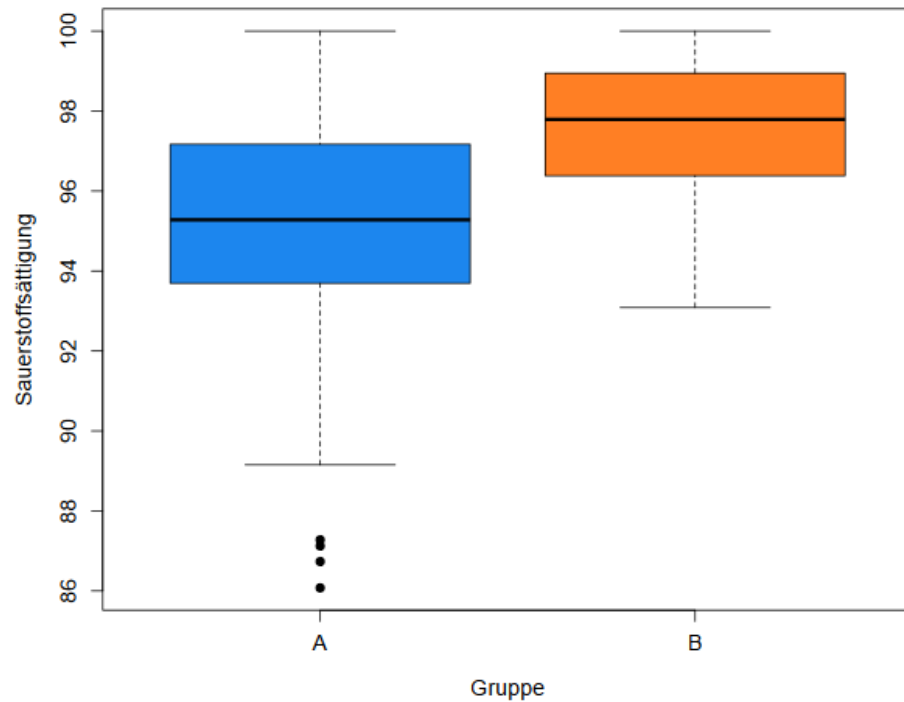


Kontingenztafeln sind auch für mehr als 2 Merkmale möglich, werden dann aber schnell unübersichtlich. Hier ein Beispiel für 3 Merkmale:

Klasse \ Überlebt	Geschlecht: m		Geschlecht: w		Randhäufigkeit
	Nein	Ja	Nein	Ja	
1	118	62	4	141	325
2	154	25	13	93	285
3	422	88	106	90	706
Crew	670	192	3	20	885
Randhäufigkeit	1364	367	126	344	2201

- Zusammenhang zwischen einem nominalskalierten und einem metrischen Merkmal

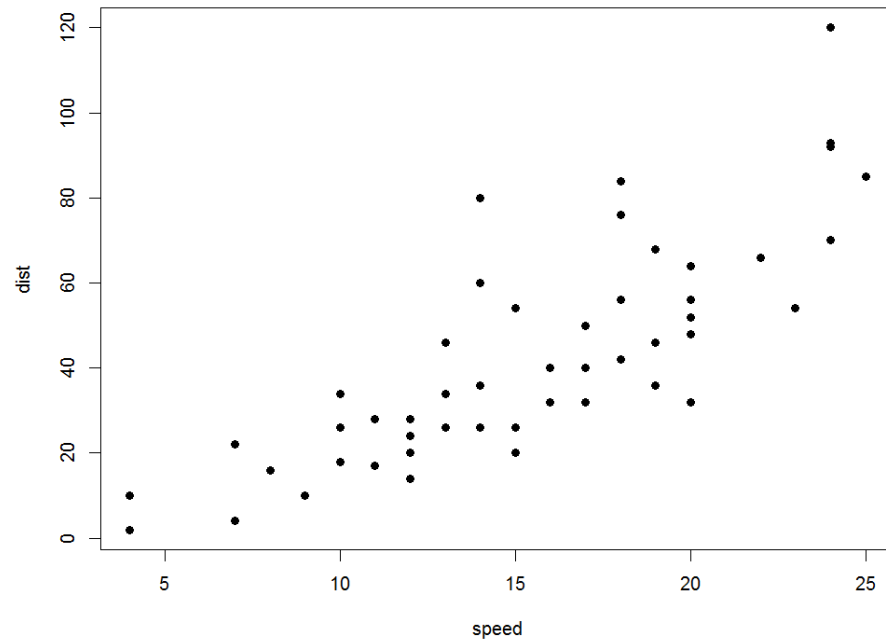
Der Zusammenhang zwischen einem nominalskalierten und einem metrischen Merkmal läßt sich sehr gut anhand von mehreren Boxplots darstellen. Hierbei wird für jede Ausprägung des nominalskalierten Merkmals ein Boxplot (anhand der Daten aus dem metrischen Merkmal) gezeichnet: In einer Studie wurde die Sauerstoffsättigung von zwei Patientengruppen miteinander verglichen.



Hierbei ist das nominalskalierte Merkmal die Patientengruppe (A oder B) und das metrische Merkmal ist die Sauerstoffsättigung.

- Zusammenhang zwischen zwei metrischen Merkmalen

Den Zusammenhang zwischen zwei metrischen Merkmalen kann man gut anhand eines einfachen Streudiagramms visualisieren. Nachfolgendes Bild zeigt den Zusammenhang zwischen der Geschwindigkeit eines Autos und des Bremsweges (historische Daten von 50 Versuchsfahrten aus den 1920er Jahren).



3 Supervised Learning (I): parameterbasierte Verfahren

3.1 Einführung

Ziel: Schätzen eines Zusammenhangs zwischen Einflussvariablen und einer Zielvariable

Einflussvariablen $\longrightarrow$ Zielvariable

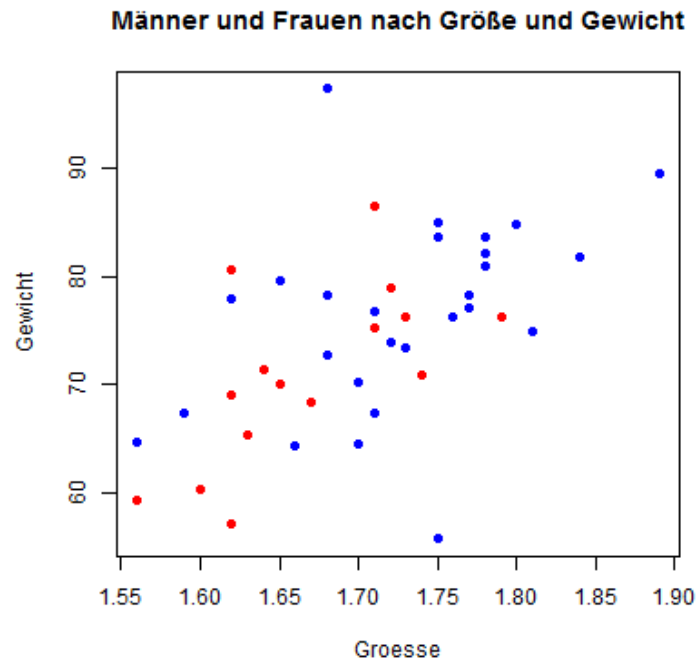
Einflussvariablen:

- eine oder mehrere Einflussvariablen möglich
- Einflussvariablen können sowohl nominal als auch metrisch sein

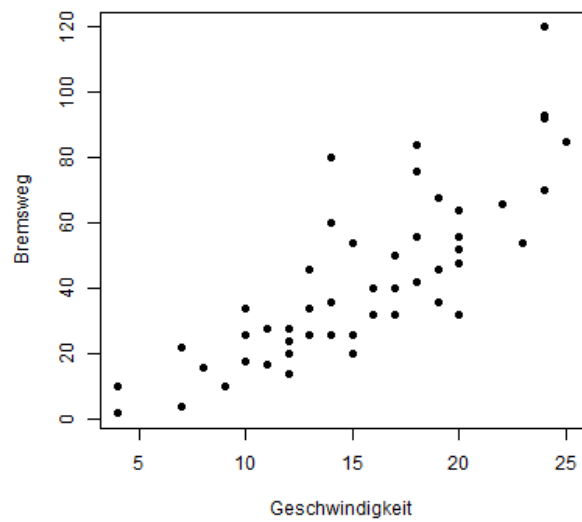
Zielvariable:

- immer nur eine Zielvariable möglich
- Zielvariable kann nominal oder metrisch sein:
 - nominal: Klassifikation
 - metrisch: Regression

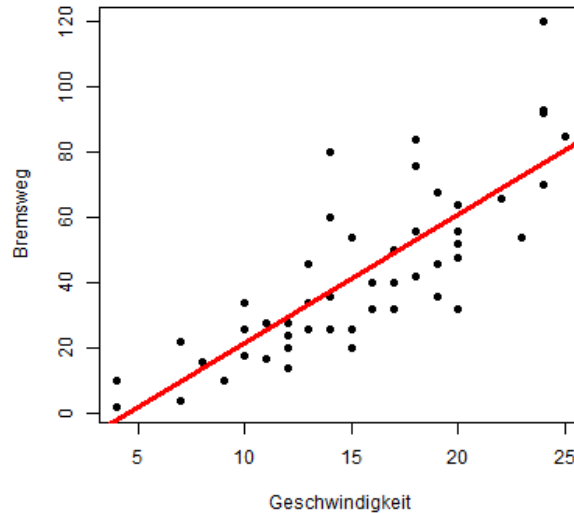
Beispiel für Klassifikation: Geschlecht in Abhängigkeit von Größe und Gewicht



Beispiel für Regression: Bremsweg in Abhängigkeit von der Geschwindigkeit



Der tendenzielle Zusammenhang zwischen Geschwindigkeit und Bremsweg wird durch die Regressionsgerade beschrieben (die rote Linie im Bild):



Die Geschwindigkeit ist hier in Meilen pro Stunde (mph) gemessen und der Bremsweg ist in Fuß (ft) gemessen. Die Steigung der Regressionsgerade (rote Linie) in dem Bild beträgt 3,9 ft/mph. Das heißt: Wenn man die Geschwindigkeit um eine Meile pro Stunde erhöht, dann verlängert sich im Durchschnitt der Bremsweg um 3,9 Fuß. Umgerechnet auf das internationale Einheitensystem bedeutet das: Wenn man die Geschwindigkeit um einen Kilometer pro Stunde erhöht, dann verlängert sich im Durchschnitt der Bremsweg um 74,5 cm.

Hinweis: Hierbei handelt es sich um historische Daten aus den 1920er Jahren. Die Geschwindigkeiten sind hier entsprechend niedrig. Bei höheren Geschwindigkeiten ist der Bremsweg wesentlich länger.

3.2 Regression

Wir behandeln zunächst die Regression und erst danach die Klassifikation. Die Regression ist leichter verständlich und anschaulicher als die Klassifikation.

3.2.1 Einfache lineare Regression

Der anschaulichste Spezialfall einer Regression ist die sogenannte einfache lineare Regression. Hierbei ist die Zielvariable metrisch und es gibt nur eine Einflussvariable, die ebenfalls metrisch ist.

Beispiel: Bremswegdaten

Wir haben hier 50 Datenpunkte $(x_1, y_1), (x_2, y_2), \dots, (x_{50}, y_{50})$, wobei

x_i = Geschwindigkeit (in mph) bei Fahrt Nr. i

y_i = Bremsweg (in ft) bei Fahrt Nr. i

Das Regressionsmodell lautet hier

$$\text{Bremsweg} = \beta_0 + \beta_1 \cdot \text{Geschwindigkeit} + \text{Zufallsabweichung}$$

bzw. in etwas genauerer Formelschreibweise

$$y_i = \beta_0 + \beta_1 \cdot x_i + \varepsilon_i,$$

wobei ε_i die Zufallsabweichung von der Regressionsgerade bei der i -ten Fahrt bezeichnet. Wäre die Zufallsabweichung stets gleich 0 (also $\varepsilon_i = 0$), dann würden alle Punkte genau auf einer Linie liegen (nämlich auf der Regressionsgeraden).

Die beiden Parameter β_0 und β_1 beschreiben die Regressionsgerade; dabei ist β_0 der Intercept (Achsenabschnitt) und β_1 die Steigung der Regressionsgeraden. Die Parameter β_0 und β_1 werden aus den Daten mit der Methode der kleinsten Quadrate berechnet:

$$\text{Finde } \beta_0, \beta_1, \text{ so dass } \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_i))^2 \text{ minimal ist.}$$

Die Regressionsgerade ist also die Gerade, die am besten zu den Daten passt. Dabei heißt “am besten”, dass sie die Summe der quadrierten Abstände zwischen Model $\beta_0 + \beta_1 x_i$ und Realität y_i minimiert.

3.2.2 Multiple lineare Regression

Nun der allgemeinere Fall: Wir haben eine metrische Zielvariable und mehrere (metrische oder nominale) Einflussvariablen.

Das Ergebnis einer linearen Regression sind Schätzungen für Koeffizienten. Diese Koeffizienten beschreiben die Größe des Einflusses² der Einflussvariablen.

Welche Koeffizienten werden in der Regression geschätzt und was bedeuten die jeweiligen Koeffizienten:

- metrische Einflussvariable: 1 Koeffizient

Für jede metrische Einflussvariable wird ein Regressionskoeffizient geschätzt. Dieser Koeffizient beschreibt den Zusammenhang zwischen der jeweiligen Einflussvariable und der Zielvariable. Der Koeffizient besagt, um wieviele Einheiten sich die Zielvariable (im Mittel) erhöht, wenn sich die Einflussvariable um eine Einheit erhöht.

- kategorielle Einflussvariable mit k Kategorien: $k - 1$ Koeffizienten

Für eine kategorielle Einflussvariable mit k Kategorien werden $k - 1$ Koeffizienten geschätzt. Bei einer kategoriellen Einflussvariablen ist eine Kategorie immer die “Basiskategorie”. Für jede der anderen Kategorien wird jeweils ein Koeffizient geschätzt; dieser Koeffizient gibt an, um wieviel sich die Zielvariable (im Mittel) bei dieser Kategorie von der Basiskategorie unterscheidet – wenn sonst alles andere gleich ist.

- Intercept: 1 Koeffizient

Bei jeder Regression wird (einmal) ein Koeffizient für den Intercept geschätzt (vgl. den Abschnitt “Einfache lineare Regression”). Dieser Koeffizient hat (meist) keine besondere interpretatorische Bedeutung.

Beispiel: Daten zu Körpergewicht

Einflussvariablen:

- Geschlecht (kategoriell)
- Alter (metrisch)
- Größe (metrisch)

²Eigentlich ist das Wort “Einfluss” hier nicht richtig, denn es beinhaltet einen kausalen Zusammenhang, der allein durch die Berechnung einer Regression nicht nachgewiesen werden kann. Statt “Einfluss” muss man eigentlich “Zusammenhang” sagen.

Zielvariable:

- Gewicht (metrisch)

Ergebnis für die geschätzten Koeffizienten aus linearen Regression:

- Intercept: $\beta_0 = -28,8347$
- Geschlecht: $k = 2$ Kategorien, also $2-1=1$ Koeffizient

$$\beta_1 = -2.6776 \text{ (für "weiblich")}$$

Basiskategorie ist "männlich"

Somit: Frauen sind – bei gleicher Körpergröße und gleichem Alter – im Schnitt um 2,6776 kg leichter als Männer

- Alter: $\beta_2 = 0,1173$

Pro Lebensjahr nimmt das Gewicht also im Schnitt um 0,1173 kg zu.

- Größe: $\beta_3 = 58,0053$

Mit jedem Meter Körpergröße nimmt das Gewicht also im Schnitt um 58,0053 kg zu.

$$58,0053 \frac{\text{kg}}{\text{m}} = 58,0053 \frac{\text{kg}}{100 \text{ cm}} = 0,580053 \frac{\text{kg}}{\text{cm}}.$$

Also anders ausgedrückt: Mit jedem Zentimeter Körpergröße nimmt das Gewicht im Schnitt um 0,580053 kg zu.

Verwendung der Regressionskoeffizienten zur Prognoseberechnung: Wie schwer ist im Schnitt ein 1,80 m großer, 37 Jahre alter Mann:

$$\begin{array}{ll} \text{Intercept:} & -28,8347 \\ \text{Alter 37:} & 37 \cdot 0,1173 \\ \text{Größe 1,80:} & 1,8 \cdot 58,0053 \\ \text{Geschlecht m:} & 0 \end{array}$$

Rechnung: $-28,8347 + 37 \cdot 0,1173 + 1,8 \cdot 58,0053 + 0 = 79,91494$
Wenn es sich stattdessen um eine Frau handelt, dann würde die 0 für Geschlecht männlich durch den Koeffizienten $\beta_1 = -2.6776$ für weiblich ersetzt

werden. Somit ergäbe sich für eine Frau (bei gleichem Alter und gleicher Körpergröße):

$$-28,8347 + 37 \cdot 0,1173 + 1,8 \cdot 58,0053 + (-2,6776) = 77,23734$$

Bewertung von Prognosen

Prognosen sind eines der Haupteinsatzgebiete von KI-Systemen. Um die Leistungsfähigkeit so eines Systems zu bewerten, muss schon bei der Entwicklung des Systems die Genauigkeit der Prognosefunktion f systematisch berechnet werden. Dies geschieht durch Kennzahlen. Eine der am besten geeigneten Kennzahlen ist der sogenannte *Mean Absolute Error* (MAE):

$$\text{MAE}(f) = \frac{1}{n} \sum_{i=1}^n |y_i - f(x_i)|.$$

Der MAE hat eine klare und einfache Bedeutung: $\text{MAE}(f) = 6,2$ bedeutet, dass der Algorithmus den Wert der Zielvariablen durchschnittlich bis auf $\pm 6,2$ genau schätzt.

3.3 (Binäre) Klassifikation: Logistische Regression

Binäre Klassifikation:

- wie Regression, nur dass die Zielvariable nominal mit 2 Kategorien (0 und 1) ist
- Schätzung der Koeffizienten wie bei Regression, aber die Koeffizienten β_j sind nicht mehr so leicht interpretierbar
- positive Werte des Koeffizienten β_j bedeuten: je höher der Wert der entsprechenden Einflussvariablen, desto höher ist die Wahrscheinlichkeit, dass die Zielvariable den Wert “1” hat;
negative Werte des Koeffizienten β_j bedeuten: je höher der Wert der entsprechenden Einflussvariablen, desto niedriger ist die Wahrscheinlichkeit, dass die Zielvariable den Wert “1” hat.

Seien β_0 (Intercept) und $\beta_1, \beta_2, \dots, \beta_m$ die geschätzten Koeffizienten und $x_1, x_2, \dots, x_m$ konkrete Werte (Ausprägungen) der entsprechenden Einflussvariablen. Dann können hieraus Prognosen dafür berechnet werden, dass

die Zielvariable den Wert “1” hat. Berechne

$$z = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \dots \beta_m \cdot x_m$$

Dann ist die Wahrscheinlichkeit, dass die Zielvariable den Wert “1” hat, gleich

$$\frac{\exp(z)}{1 + \exp(z)} .$$

Bewertung von Prognosen

Auch im Fall einer Klassifikation, muss schon bei der Entwicklung des Systems die Genauigkeit der Prognosen systematisch berechnet werden. Eine übliche Kennzahl hierfür ist die **Misclassification Error Rate**. Bei einer (binären) Klassifikation gibt es nur zwei Möglichkeiten, nämlich $y = 0$ oder $y = 1$. Die Prognose $\hat{y} = 0$ oder $\hat{y} = 1$ ist also entweder richtig oder falsch. (Nur ein bisschen falsch wie in der Regression geht hier nicht.) Die Misclassification Error Rate zählt den prozentualen Anteil an falschen Prognosen.

4 Supervised Learning (II): Nichtparametrische Verfahren

4.1 Einführung

Maschinelle Lernverfahren funktionieren im Wesentlichen wie klassische statistische Verfahren und sind nichts grundsätzlich neues. Viele Verfahren, die heute unter dem Begriff “Maschinelle Lernverfahren” oder “selbstlernende Algorithmen”, eingeordnet werden, sind statistische Verfahren, die von Informatikern entwickelt wurden. In der Informatik werden für statistische Verfahren jedoch andere Begriffe verwendet als in der klassischen Statistik:

Bezeichnung in Statistik	Bezeichnung in Informatik/KI
Datensatz	Trainingsdaten
statistisches Verfahren z.B. logistische Regression, Support Vector Machines	Lern-Algorithmus z.B. logistische Regression, Support Vector Machines
Berechnung der Schätzung	Trainieren des Algorithmus / Lernen
Untersuchung von Zusammenhängen zwischen x und y	Mustererkennung

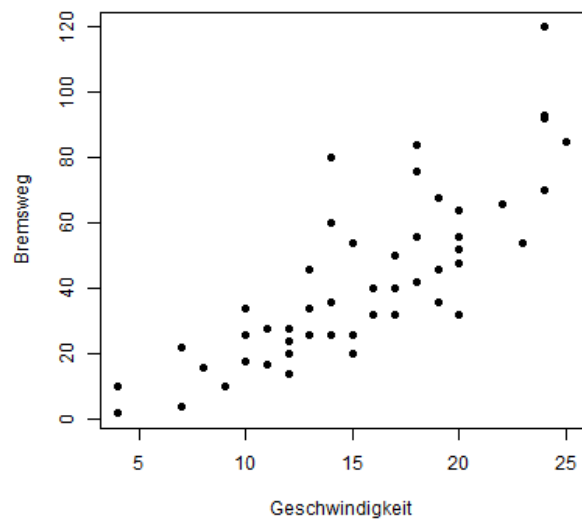
Betrachten wir wieder unser Beispiel aus dem Regressionskapitel: Bremsweg in Abhängigkeit von der Geschwindigkeit.

Es handelt sich also um ein überwachtes Lernproblem, wobei “Geschwindigkeit” die Einflussvariable ist und “Bremsweg” die Zielvariable ist.

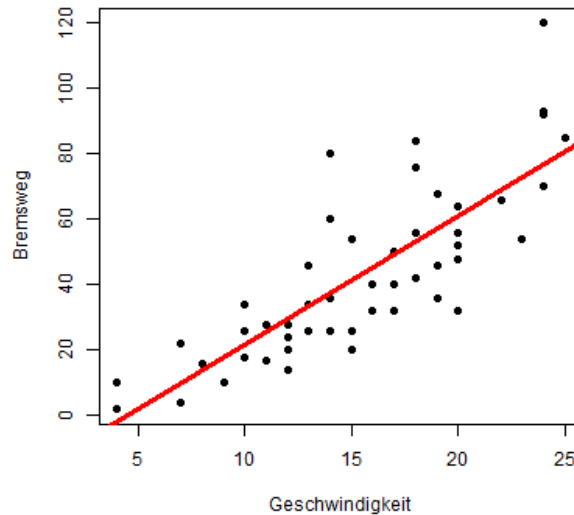
Die Daten haben die Form:

	speed	dist
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10
7	10	18
8	10	26
$\vdots$	$\vdots$	$\vdots$

Streudiagramm:



Den tendenziellen Zusammenhang zwischen Geschwindigkeit und Bremsweg kann man durch eine Regressionsgerade beschreiben:



Wenn man so möchte, dann kann man dieses lineare Regressionsverfahren als ein maschinelles Lernverfahren bezeichnen:

Der Datensatz sind unsere Trainingsdaten, das lineare Regressionsverfahren ist der Lernalgorithmus, bei der Berechnung der Regression wird das Verfahren trainiert. Die Regressionsgerade bzw. die in der Regression berechneten Regressionskoeffizienten beschreiben das in den Daten gefundene Muster.

Soweit ist das also nichts Neues. Was ist denn nun das Neue an den maschinellen Lernverfahren: Viele maschinelle Lernverfahren sind sogenannte nichtparametrische oder nichtlineare Verfahren.

Die lineare Regression ist ein parametrisches Verfahren, weil hierbei dem Verfahren bereits ein Modell vorgegeben ist, dass nur von wenigen Parametern abhängt:

Das Regressionsmodell lautet in unserem Bremsweg-Beispiel:

$$\text{Bremsweg} = \beta_0 + \beta_1 \cdot \text{Geschwindigkeit} + \text{Zufallsabweichung}$$

bzw. in etwas genauerer Formelschreibweise

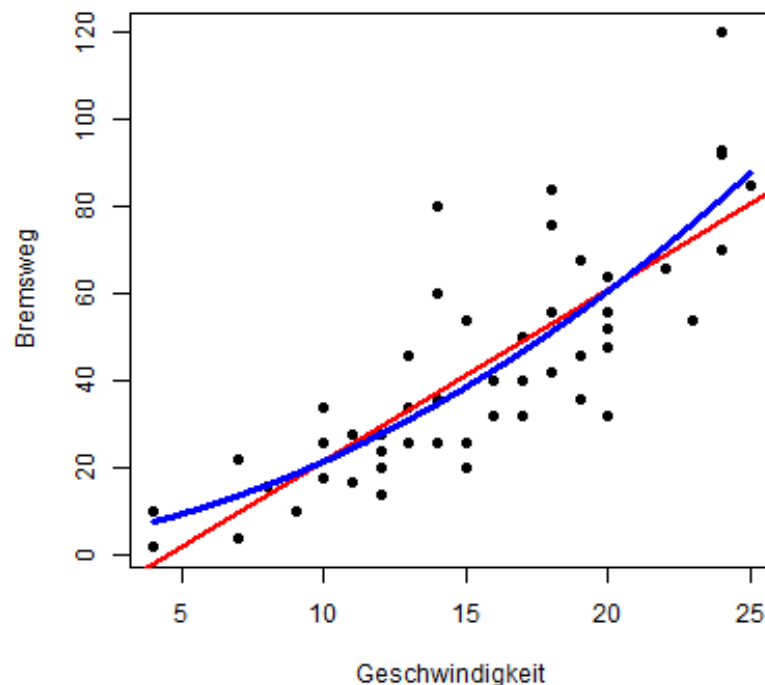
$$y_i = \beta_0 + \beta_1 \cdot x_i + \varepsilon_i,$$

wobei ε_i die Zufallsabweichung von der Regressionsgerade bei der i -ten Fahrt bezeichnet.

Hierbei ist also schon ein genaues Modell vorgegeben, das besagt, wie der Zusammenhang zwischen Geschwindigkeit und Bremsweg aussieht: ein linearer Zusammenhang. Das Modell muss dabei vorher der Datenanalyst vorgeben und wenn das Modell “falsch” ist, dann sind auch die Ergebnisse aus der linearen Regression nicht gut. Im Bremsweg-Beispiel entspricht unser Modell ja auch tatsächlich nicht der Realität. Der Zusammenhang zwischen Geschwindigkeit und Bremsweg ist in Wirklichkeit nicht linear, sondern quadratisch. Besser wäre also ein quadratisches Modell:

$$y_i = \beta_0 + \beta_1 \cdot x_i + \beta_2 \cdot x_i^2 + \varepsilon_i.$$

Die Parameter β_0 , β_1 und β_2 lassen sich hier auch mit einer linearen Regression berechnen: einfach im Datensatz eine Spalte mit der quadrierten Geschwindigkeit ergänzen und dann die lineare Regression (wie gehabt) in R berechnen. Hier ist das Ergebnis (blaue Linie):



Im obigen Text steht bzgl. dem Modell das Wort “falsch” in Anführungszeichen. Es ist nämlich nicht die Aufgabe eines Modells “richtig” zu sein in dem Sinn, dass das Modell die Realität exakt wiedergibt. Jedes Modell ist nur eine Vereinfachung der Realität. In unserem Bremsweg-Beispiel ist die lineare Kurve zwar “falsch” aber - zumindest bei den niedrigen Geschwindigkeiten - bereits eine sehr gute (für viele Zwecke ausreichende) Beschreibung des Zusammenhangs. Der Statistiker George Box schrieb hierzu: “All models are wrong; some models are useful”.

Wie findet man aber nun ein gutes (“nützliches”) Modell, das die Realität genügend gut beschreibt? Hierzu gibt es insbesondere drei Möglichkeiten:

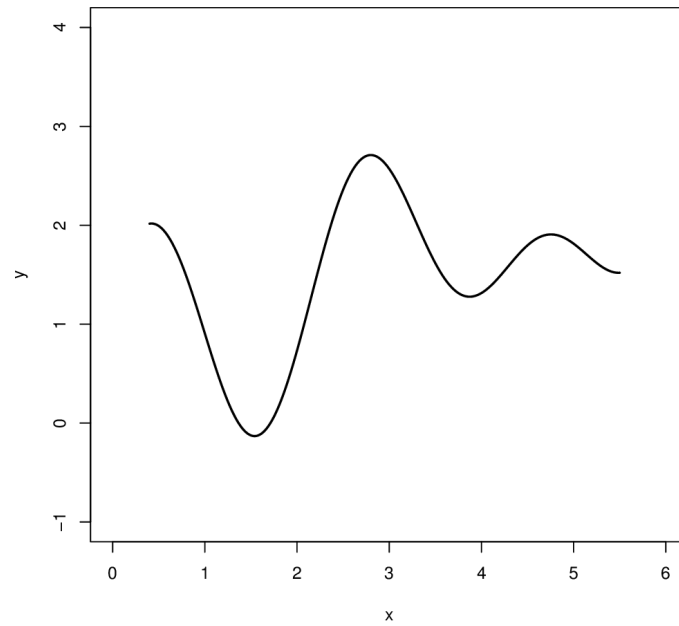
1. durch Anschauen der Daten (schwierig bei mehreren Einflussvariablen),
2. durch physikalische Überlegungen oder
3. durch die Wahl eines nichtparametrischen Verfahrens.

Nichtparametrische Verfahren sind statistische Verfahren, die keine Modellvorgabe benötigt und das Modell eigenständig aus den Daten berechnen. Es handelt sich hierbei also um eine Regression der Form

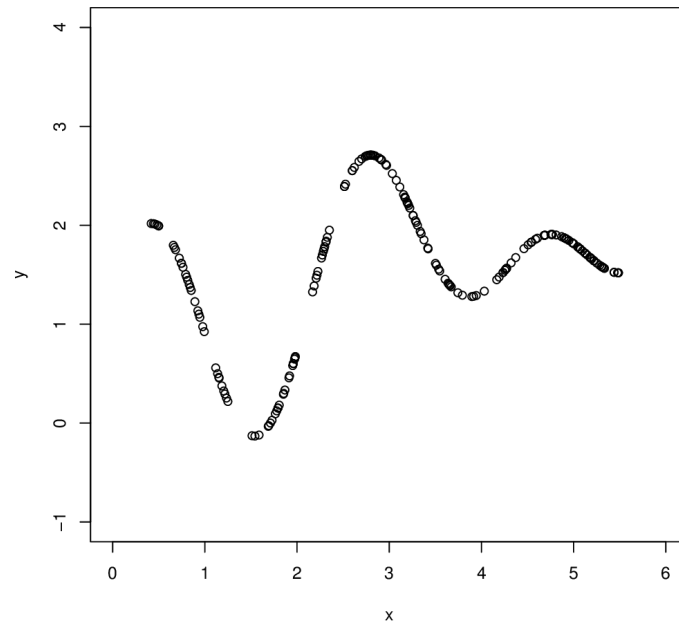
$$y_i = f(x_i) + \varepsilon_i,$$

wobei die Funktion f irgendeine unbekannte Funktion ist, ohne dass die Form der Funktion (linear, quadratisch etc.) vorgegeben ist. Die Funktion f wird von nichtparametrischen Verfahren rein aus den Daten berechnet. Maschinelle Lernverfahren sind oft nichtparametrische Verfahren und bei der Verwendung solcher Verfahren muss man ein paar Dinge beachten, die bei parametrischen Verfahren (z.B. lineare Regression) noch keine Rolle gespielt haben.

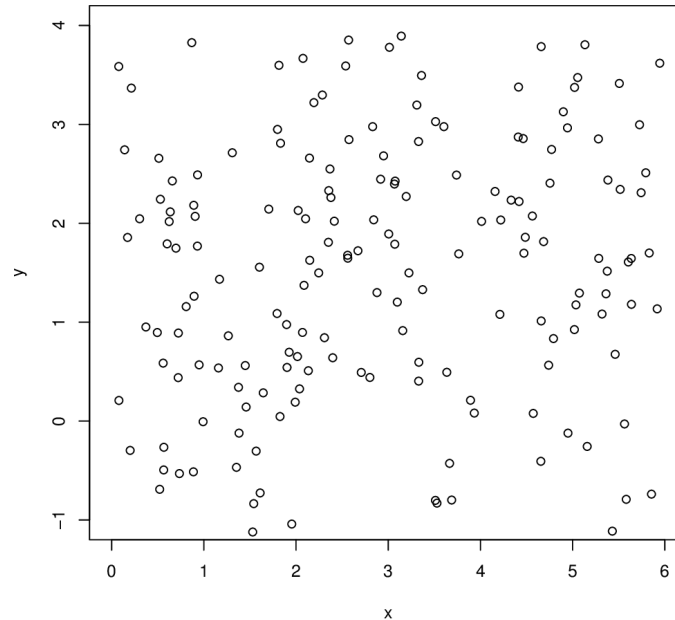
Folgendes Bild zeigt den theoretischen Zusammenhang zwischen zwei Variablen x und y :



In einer idealen Welt ohne Messfehler und ohne Zufallsabweichungen könnte man Daten erheben, die genau dem theoretischen Zusammenhang folgen würden. Die erhobenen Daten würden dann etwa so aussehen:



Hieraus würde sich der genaue theoretische Zusammenhang leicht rekonstruieren lassen. In der realen Welt mit Messfehlern und Zufallsabweichungen sehen die Daten aber eher so aus:



Wie lässt sich der Zusammenhang zwischen den Variablen x und y nun aus solchen Daten (mit Messfehlern und Zufallsabweichungen) rekonstruieren – ohne Kenntnis irgendeines passenden Modells?

1. Idee: Gehe vor wie bei linearer Regression, nur dass nun jede beliebige Funktion f zugelassen ist:

- Lineare Regression: Modell z.B. wie beim Bremsweg

$$f_{\beta}(x) = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2.$$

Suche nun Werte für β_0 , β_1 und β_2 , die am besten zu den Daten passen, d.h. der (quadratische) Abstand zwischen Modell und Daten ist minimal:

$$\text{Minimiere} \quad \left(y_1 - f_{\beta}(x_1)\right)^2 + \dots + \left(y_n - f_{\beta}(x_n)\right)^2$$

- Nichtparametrische Regression: kein Modell vorhanden

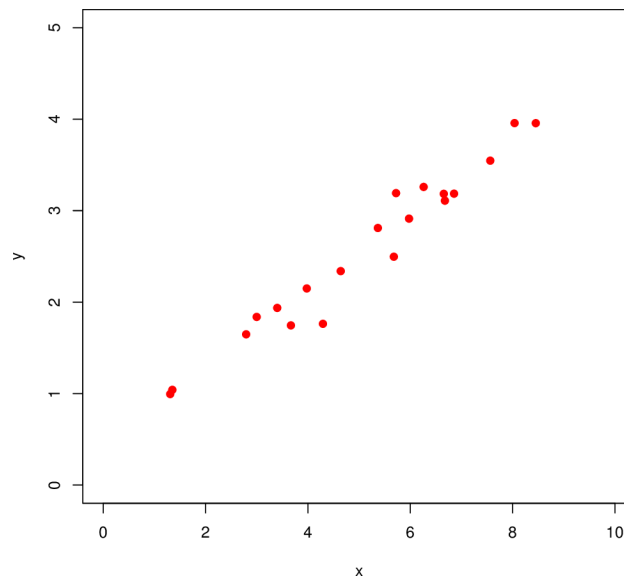
$$f(x) = ???$$

Suche nun nach einer beliebigen Funktion f , die am besten zu den Daten passt, d.h. der (quadratische) Abstand zwischen der Funktion f und den Daten ist minimal:

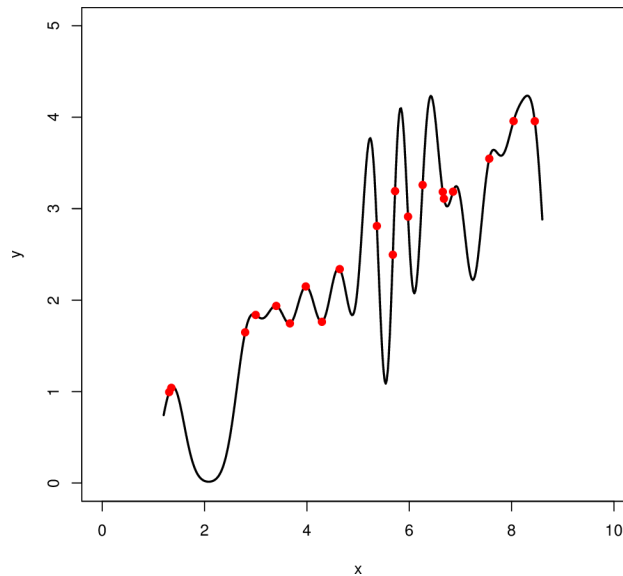
$$\text{Minimiere} \quad \left(y_1 - f(x_1)\right)^2 + \dots + \left(y_n - f(x_n)\right)^2$$

Wenn man das macht, dann hat man aber das Problem des Overfittings: Die vom Algorithmus errechnete Funktion passt “zu gut” zu den Daten, beschreibt aber keinen allgemeinen Zusammenhang.

Dies sieht man gut an folgendem Datenbeispiel:



Hier folgen die Daten in etwa einem linearen Trend. Geht man aber wie in der 1. Idee beschrieben vor, erhält man keine lineare Funktion f , sondern folgendes unsinniges Ergebnis:



Die nach der 1. Idee berechnete Funktion ist unsinnig, weil sie für die Daten zu kompliziert ist.

Man muss daher einen Kompromiss finden zwischen

- Wie gut soll die Funktion f die Daten beschreiben?
- Wie kompliziert darf die Funktion f sein?

Je besser die Funktion die Daten beschreibt, desto komplizierter wird die Funktion. Bei vielen Verfahren / Algorithmen wird daher die Komplexität der Funktion “bestraft”, d.h. einfache Funktionen werden bevorzugt:

2. Idee: Gehe vor wie bei linearer Regression, nur dass nun jede beliebige Funktion f zugelassen ist, aber einfache Funktionen bevorzugt werden:

- Lineare Regression: Modell z.B. wie beim Bremsweg

$$f_{\beta}(x) = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2.$$

Suche nun Werte für β_0 , β_1 und β_2 , die am besten zu den Daten passen, d.h. der (quadratische) Abstand zwischen Modell und Daten ist minimal:

$$\text{Minimiere} \quad \left(y_1 - f_{\beta}(x_1)\right)^2 + \dots + \left(y_n - f_{\beta}(x_n)\right)^2$$

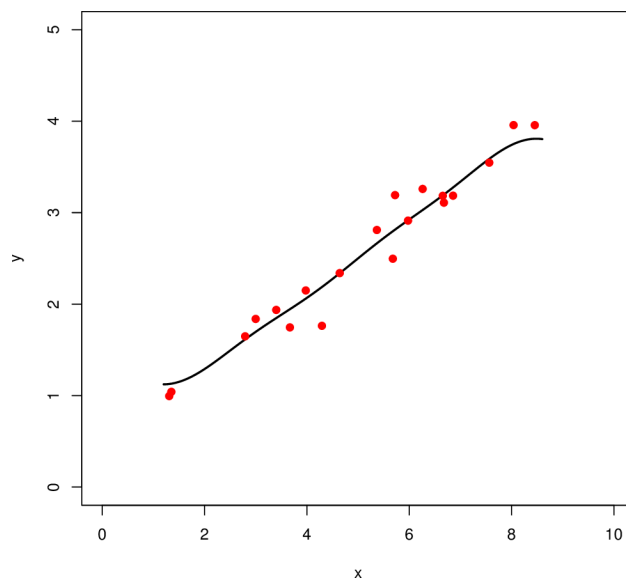
- Nichtparametrische Regression: kein Modell vorhanden

$$f(x) = ???$$

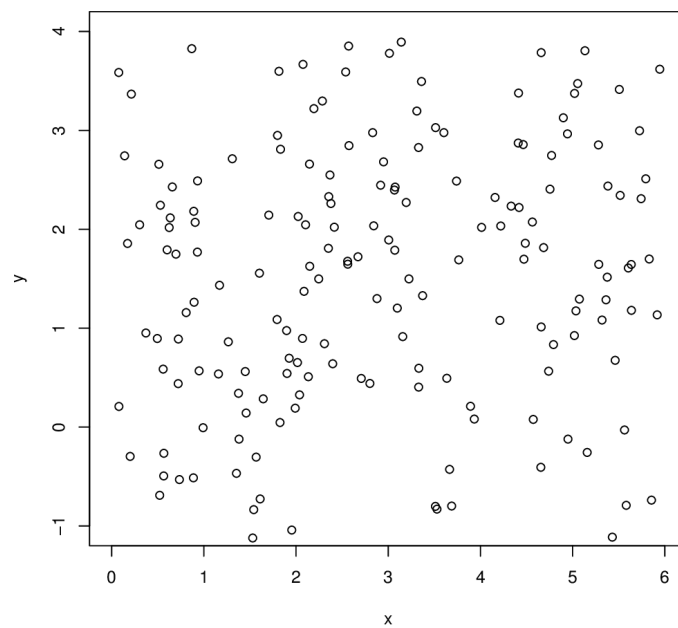
Suche nun nach einer beliebigen Funktion f , die einen möglichst guten Kompromiss zwischen den Daten und der Komplexität darstellt, d.h. die Kombination aus dem (quadratische) Abstand zwischen der Funktion f und den Daten mit der Komplexität der Funktion f ist minimal:

$$\text{Minimiere } \left(y_1 - f(x_1)\right)^2 + \dots + \left(y_n - f(x_n)\right)^2 + \text{Komplexität}(f)$$

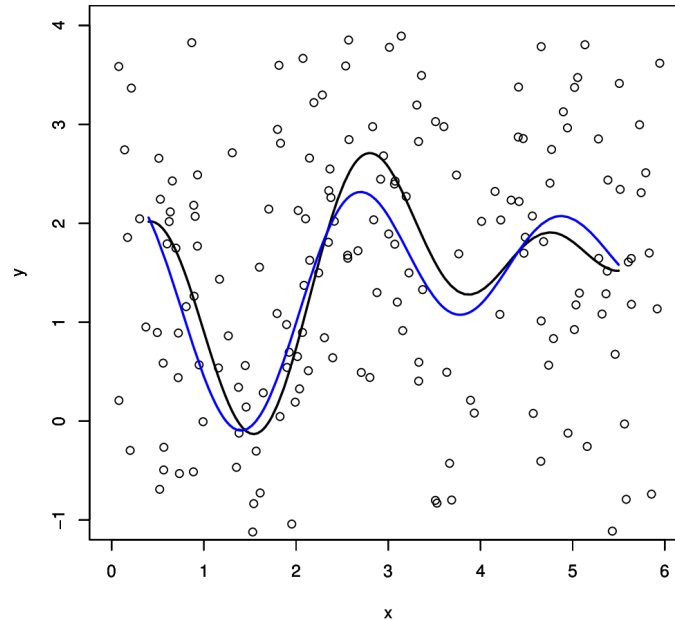
Dieses Vorgehen löst das Problem des Overfittings. Als Ergebnis erhält man für die Beispieldaten:



Sofern genügend Daten vorhanden sind, kann mit solchen Verfahren nun jeder beliebige Zusammenhang aus den Daten errechnet werden. Betrachten wir noch einmal folgende Beispieldaten mit Messfehlern und Zufallsabweichungen vom theoretischen Zusammenhang zwischen zwei Variablen x und y :



Mit dem Vorgehen aus der 2. Idee lässt sich der unbekannte theoretischen Zusammenhang (schwarze Linie) recht gut aus den Daten rekonstruieren (blaue Linie):



4.2 Entscheidungsbäume

Fallbeispiel: In einem meiner Unternehmensprojekte sollten Untersuchungen zur Entwicklung eines entscheidungsunterstützenden Systems bei der Herstellung von Glaswolle durchgeführt werden. Konkret ging es darum, Muster zwischen den Einstellungen der Produktionsanlagen und der Qualität der produzierten Glaswolle zu finden. Eine Vorgabe war, dass die von den Algorithmen identifizierten Muster für den Anwender nachvollziehbar sein sollten. Bei der Problemstellung handelt es sich um ein überwachttes Lernproblem:

Einstellungen der Produktionsanlagen $\longrightarrow$ Qualität der Glaswolle

Die meisten maschinellen Lernverfahren wie Support Vector Regression oder Neuronale Netze eignen sich hier nicht, weil sie Black-Box-Lösungen liefern: die von den Algorithmen identifizierten Muster sind für den Anwender nicht nachvollziehbar. Dies liegt daran, dass die Algorithmen eine beliebig komplexe Prognosefunktion

$$f : \mathbb{R}^m \rightarrow \mathbb{R}, \quad (x_1, x_2, \dots, x_m) \mapsto f(x_1, x_2, \dots, x_m)$$

berechnen. Diese Funktion f ist bei diesen Verfahren nicht zugänglich für Interpretationen und kann – aufgrund der hohen Dimensionalität – nicht grafisch dargestellt werden. Hat man z.B. $m = 10$ Einflussvariablen, bräuchte man ein Bild in einem 11-dimensionalen Raum. Es gibt aber auch maschinelle Lernverfahren, die nachvollziehbare Muster erzeugen. Dies ist möglich, indem man sich auf bestimmte Prognosefunktionen f beschränkt. Ein solches Verfahren sind Entscheidungsbäume. Bei Entscheidungsbäumen beschränkt man sich auf Funktionen der Gestalt

$$f(x) = c_1 \mathbb{1}_{R_1}(x) + c_2 \mathbb{1}_{R_2}(x) + \dots c_T \mathbb{1}_{R_T}(x)$$

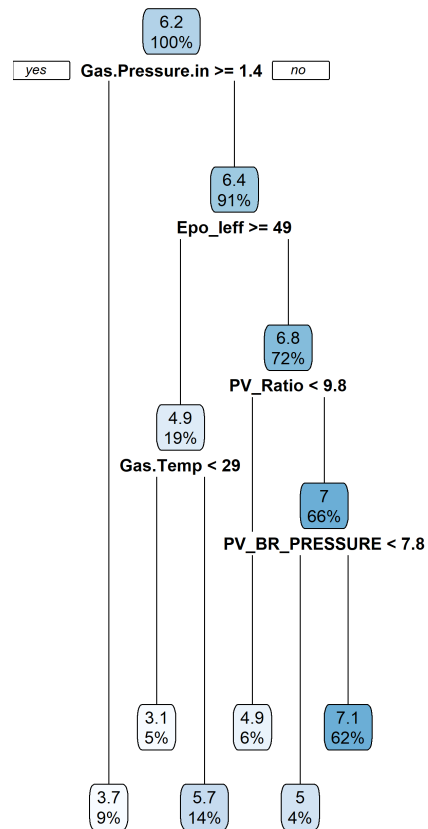
Hierbei ist jeweils R_ℓ ein Rechteck im $\mathbb{R}^m$ und

$$\mathbb{1}_{R_\ell}(x) = 1 \text{ falls } x \in R_\ell \quad \text{und} \quad \mathbb{1}_{R_\ell}(x) = 0 \text{ falls } x \notin R_\ell.$$

Wenn sich die Rechtecke nicht überlappen, dann kann man die Funktion auch so darstellen:

$$f(x) = \begin{cases} c_1 & \text{falls } x \in R_1 \\ \vdots & \vdots \\ c_\ell & \text{falls } x \in R_\ell \\ \vdots & \vdots \\ c_T & \text{falls } x \in R_T \end{cases}$$

Diese Funktionen lassen sich sehr schön grafisch durch Bäume darstellen. Folgendes Bild zeigt einen möglichen Entscheidungsbaum aus dem Projekt zur Produktion von Glaswolle



Entscheidungsbaum zur Einstellung der Anlage bei der Produktion von Glaswolle

Hierbei sind “Gas.Pressure.in”, “Epo_levf”, “Gas.Temp” etc. die Einstellparameter der Produktionsanlage (Einflussvariablen). Die Dezimalzahlen in den Kästchen sind genau die Werte c_ℓ der Prognosefunktion f . Dabei handelt es sich also um die jeweilige Prognose für die produzierte Qualität (Zielvariable). Die Qualität wird mit einer Kennzahl zwischen 1 (niedrigste Qualität) und 10 (höchste Qualität) gemessen. Die Prozentzahlen geben noch ergänzend an, wieviele Datenpunkte in den jeweiligen Bereich (also das entsprechende Rechteck R_ℓ) fallen. Das heißt: Wenn der Gasdruck (“Gas.Pressure.in”) den Schwellenwert 1,4 erreicht oder überschreitet, dann ist die Prognose für die Qualität gleich 3,7 (ziemlich niedrige Qualität). Im Trainingsdatensatz wurde der Schwellenwert 1,4 für den Gasdruck in 9% der Fälle überschritten.

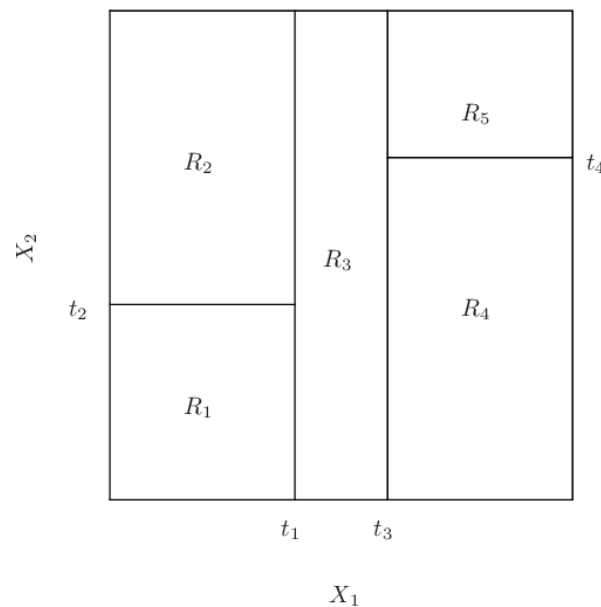
Frage: Wie sollte die Anlage eingestellt werden, um eine höchstmögliche Qua-

lität zu erreichen?

Wenn $x_1, x_2, \dots, x_m$ die verschiedenen Einflussvariablen sind, dann werden die Rechtecke durch Schnitte an einem Schwellenwert t_ℓ gebildet. Dabei wird der Raum $\mathbb{R}^m$ bei jedem Schnitt am Schwellenwert t_ℓ in zwei Hälften geteilt:

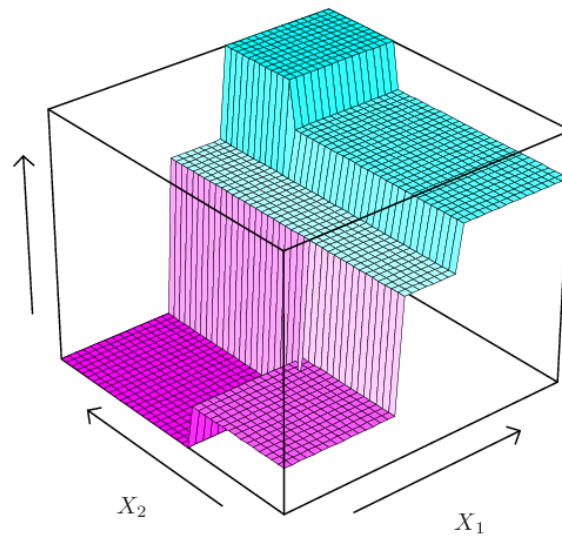
$$x_j \leq t_\ell \quad \text{und} \quad x_j > t_\ell.$$

Das nachfolgende Bild zeigt für $m = 2$ Einflussvariablen x_1 und x_2 , wie durch solche Schnitte an Schwellenwerten t_ℓ der Raum $\mathbb{R}^2$ in rechteckige Regionen R_ℓ aufgeteilt wird³:

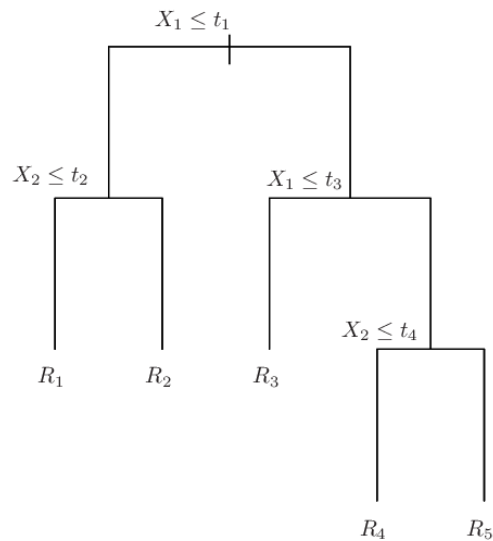


Eine hieraus resultierende Prognosefunktion f kann in diesem einfachen Fall in einem 3-dimensionalen Bild gemalt werden (weil wir ja nur $m = 2$ Einflussvariablen haben). Die Höhen der “Treppenstufen” stellen die Werte c_ℓ der Prognosefunktion f dar.

³Die nachfolgenden drei Bilder sind aus “An Introduction to Statistical Learning, with applications in R” (Springer, 2013) mit Erlaubnis der Autoren: G. James, D. Witten, T. Hastie and R. Tibshirani



Aufgrund der besonderen Gestalt lässt sich diese Prognosefunktion f auch als Baumdiagramm darstellen. Das nachfolgende Bild zeigt das Baumdiagramm für dieses Bild:



Jeder Schnitt im Raum $\mathbb{R}^m$ entspricht im Entscheidungsbaum einem Knotenpunkt.

Das Baumdiagramm hat zwei Vorteile:

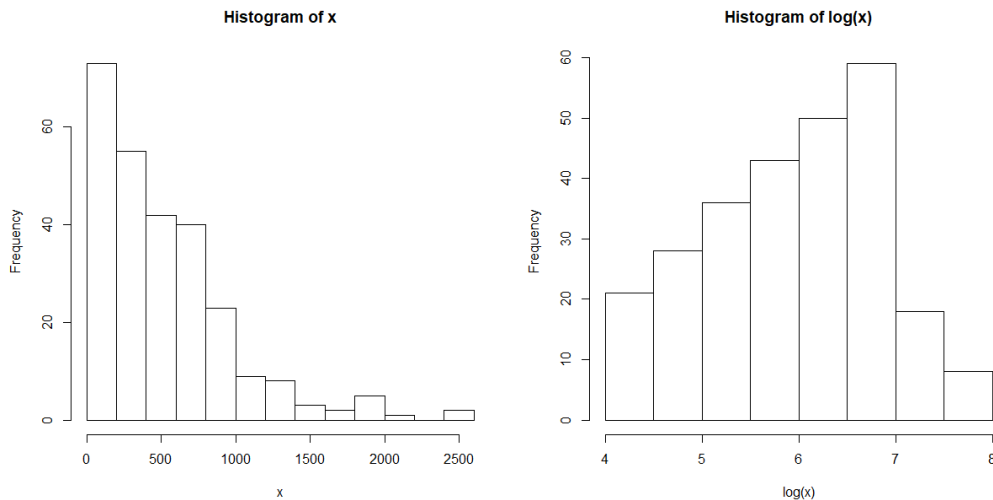
- Das Baumdiagramm kann auch für beliebig viele Einflussvariablen m gezeichnet werden.
- Das Baumdiagramm kann auch von Data-Mining-Laien problemlos interpretiert werden.

Den zweiten der beiden Vorteile sehen wir im folgenden Beispiel.

Beispiel: Spielergehälter im Baseball

Der Datensatz **Hitters** enthält Daten zu Baseballspielern. Neben dem Jahresgehalt (**Salary**), ist unter anderem auch aufgeführt, wie lange der Spieler schon in den höchsten Ligen spielt (**Years**) und wieviele Hits⁴ er im letzten Jahr hatte (**Hits**).

Mit Hilfe eines Entscheidungsbaums kann nun dargestellt werden, wie die Zielvariable **Salary** von den Einflussvariablen **Year** und **Hits** abhängt. Vor der Berechnung eines Entscheidungsbaum wurde das Gehalt (**Salary**) logarithmiert, um eine etwas symmetrischere Verteilung zu erhalten:



Das nachfolgende Bild⁵ zeigt einen entsprechenden Entscheidungsbaum:

⁴“Als Hit gilt im Baseball, wenn der Batter (Schlagmann) nach einem erfolgreichen Treffer des Balls eine Base erreicht. In der Baseballstatistik erhält der Batter einen Hit, wenn er nach einem gültigen Schlag des Balls ohne den Vorteil eines Errors oder Fielder’s Choice die erste Base erreicht.” (Wikipedia)

⁵Aus “An Introduction to Statistical Learning, with applications in R” (Springer, 2013) mit Erlaubnis der Autoren: G. James, D. Witten, T. Hastie and R. Tibshirani



Berechnung eines Entscheidungsbaumes

Um einen Entscheidungsbaum zu berechnen, sind zunächst zwei wesentliche Punkte von Bedeutung:

1. Der von den Einflussvariablen $x_1, x_2, \dots, x_m$ aufgespannte Raum $\mathbb{R}^m$ muss in Rechtecke $R_1, R_2, \dots, R_T$ aufgeteilt werden.
2. Für jedes Rechteck R_ℓ muss ein Prognosewert c_ℓ bestimmt werden.

Wir betrachten diese beiden Punkte zunächst am Beispiel der Regression:

Im Fall der Regression mit Entscheidungsbäumen wird typischerweise als Gütekriterium die Summe der quadrierten Abweichungen der Prognosen von den realen Daten (Kleinste-Quadrate-Verlustfunktion) verwendet. Hieraus ergibt sich, dass als Prognosewert c_ℓ das arithmetische Mittel optimal ist. Das heißt, zur Berechnung von c_ℓ nimmt man alle Trainingsdaten, die in das Rechteck R_ℓ fallen, und wählt als c_ℓ das arithmetische Mittel der Werte der Zielvariablen y_i von diesen Trainingsdaten.

Die Aufteilung von $\mathbb{R}^m$ in Rechtecke $R_1, R_2, \dots, R_T$ geschieht rekursiv. Dabei wird in jedem Schritt eine Einflussvariable x_j und ein Schwellenwert t

ausgewählt, so dass sich die Summe der quadrierten Abweichungen der Prognosen von den realen Daten in jedem Schritt bestmöglich verringert.

Im Fall der (binären) Klassifikation (mit einer 0-1-Zielvariablen) ist das Vorgehen ähnlich. Das arithmetische Mittel über die Werte der Zielvariablen y_i ist dann genau der Anteil $\hat{p}_\ell$ der Trainingsdaten im Rechteck R_ℓ mit Wert $y_i = 1$. Der einzige Unterschied zur Regression ist, dass als Gütekriterium nicht die Summe der quadrierten Abweichungen der Prognosen verwendet wird. Stattdessen ist z.B. der sogenannte Gini-Index

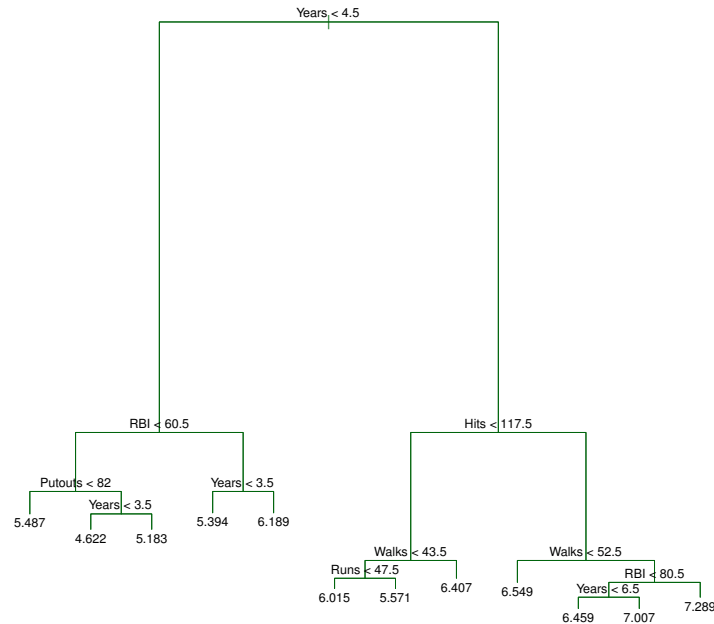
$$\hat{p}_\ell \cdot (1 - \hat{p}_\ell)$$

gebräuchlich. Der Gini-Index ist umso kleiner (gut), je homogener die Trainingsdaten im Rechteck R_ℓ sind, also wenn entweder (fast) alle Werte $y_i = 0$ oder (fast) alle Werte $y_i = 1$ sind. Der Gini-Index ist umso größer (schlecht), je inhomogener die Trainingsdaten im Rechteck R_ℓ sind.

Vermeiden von Overfitting: Regularisierung

Wie in der Einleitung zum maschinellen Lernen besprochen, muss jedes maschinelle Lernverfahren einen Kompromiss finden: Einerseits soll die Prognosefunktion gut zu den Daten passen, andererseits soll die Prognosefunktion nicht überkomplex sein. Dies wird häufig durch eine Regularisierung gelöst, die die Komplexität bestraft. Dies ist auch bei Entscheidungsbäumen der Fall. Ein gutes Maß für die Komplexität eines Entscheidungsbaumes ist die Zahl T der Endknoten des Baumes. Die Endknoten des Entscheidungsbaumes entsprechen genau den Rechtecken R_ℓ . Je mehr Endknoten ein Entscheidungsbaum hat, desto komplexer ist der Entscheidungsbaum. Das folgende Bild⁶ zeigt den Entscheidungsbaum für den Datensatz **Hitters**, wenn die Zahl der Endknoten nicht beschränkt wird. Dabei sieht man übrigens auch, dass eine Einflussvariable (z.B. **Years**) problemlos auch mehrfach ausgewählt werden kann.

⁶Aus “An Introduction to Statistical Learning, with applications in R” (Springer, 2013) mit Erlaubnis der Autoren: G. James, D. Witten, T. Hastie and R. Tibshirani



Wird die Zahl der Endknoten nicht begrenzt, dann entstehen zu komplexe Entscheidungsbäume, die zum Overfitting führen. Während der anfangs gezeigte (optimale) Entscheidungsbaum nur $T = 3$ Endknoten hatte, besitzt dieser Entscheidungsbaum $T = 12$ Endknoten.

Eine solches Overfitting wird beim Entscheidungsbaum mit Hilfe einer Regularisierung ähnlich wie bei Support Vector Machines verhindert. Anstelle der Summe der quadrierten Abweichungen der Prognosen wird

$$\text{Kennzahl}(f) = \text{Risiko}(f) + \underbrace{\lambda \cdot \text{Komplexität}(f)}_{\text{Regularisierung}}$$

minimiert. Dabei ist in der Regression das Risiko gleich

$$\begin{aligned} \text{Risiko}(f) &= \frac{(f(x_{1,1}, x_{1,2}, \dots, x_{1,k}) - y_1)^2 + \dots + (f(x_{n,1}, x_{n,2}, \dots, x_{n,k}) - y_n)^2}{n} \\ &= \text{mittlere quadratische Abweichung der Prognosen von} \\ &\quad \text{den tatsächlichen Daten} \end{aligned}$$

und das Maß für die Komplexität ist

$$\text{Komplexität}(f) = \text{Anzahl } T \text{ der Endknoten des Entscheidungsbaumes } f.$$

Mit dem Parameter λ wird gesteuert, wie stark regularisiert werden soll, d.h. wie stark die Komplexität der Funktion f bestraft werden soll. (Einfache Entscheidungsbäume mit wenigen Endknoten werden bevorzugt!)

4.3 Neuronale Netze

Betrachten wir wieder ein überwachtes Lernproblem, also eine Regression oder eine Klassifikation. Dabei haben wir wieder

- Einflussvariablen: $x_1, x_2, \dots, x_m$
- Zielvariable: y

Neuronale Netze kann man als Verallgemeinerung der linearen Regression (bzw. auch der logistischen Regression) betrachten:

Lineare Regression

Modell:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_m x_m + \varepsilon$$

Neuronale Netze

Für ein neuronales Netz werden zunächst sogenannte “abgeleitete Einflussvariablen” gebildet:

$$\begin{aligned} z_1 &= \sigma(w_{0,1} + w_{1,1}x_1 + w_{2,1}x_2 + \dots + w_{m,1}x_m) \\ z_2 &= \sigma(w_{0,2} + w_{1,2}x_1 + w_{2,2}x_2 + \dots + w_{m,2}x_m) \\ &\vdots \\ z_{m_1} &= \sigma(w_{0,m_1} + w_{1,m_1}x_1 + w_{2,m_1}x_2 + \dots + w_{m,m_1}x_m) \end{aligned}$$

Das Modell lautet dann:

$$y = g(\beta_0 + \beta_1 z_1 + \beta_2 z_2 + \dots + \beta_{m_1} z_{m_1}) + \varepsilon$$

Folgendes Bild zeigt nochmal das Modell mit den üblichen Bezeichnungen:

Abgeleitete Einflussvariablen:

$$\begin{array}{lcl}
 \text{Hidden Units} \rightarrow z_1 & = & \sigma(w_{0,1} + w_{1,1}x_1 + w_{2,1}x_2 + \dots + w_{m,1}x_m) \\
 \rightarrow z_2 & = & \sigma(w_{0,2} + w_{1,2}x_1 + w_{2,2}x_2 + \dots + w_{m,2}x_m) \\
 \vdots & & \vdots \\
 \rightarrow z_{m_1} & = & \sigma(w_{0,m_1} + w_{1,m_1}x_1 + w_{2,m_1}x_2 + \dots + w_{m,m_1}x_m)
 \end{array}
 \quad \left. \begin{array}{l} \text{Aktivierungsfunktion} \\ \text{Gewichte} \end{array} \right\} \text{1 Hidden Layer}$$

Modell:

$$y = g(\beta_0 + \beta_1 z_1 + \beta_2 z_2 + \dots + \beta_{m_1} z_{m_1}) + \varepsilon$$

Ausgabefunktion

Würde man als Aktivierungsfunktion σ die Identitätsfunktion $\sigma(x) = x$ und als Ausgabefunktion g ebenfalls die Identitätsfunktion $g(t) = t$ wählen, so würde man genau wieder die lineare Regression erhalten. (Die logistische Regression erhält man als Spezialfall, wenn man als Aktivierungsfunktion σ die Identitätsfunktion $\sigma(x) = x$ und als Ausgabefunktion g die sogenannte Sigmoid-Funktion wählt.)

Das Modell lässt sich auch in einer **Kompakteren Schreibweise** aufschreiben:

$$X := (1, x_1, x_2, \dots, x_m)^T, \quad \beta := (\beta_0, \beta_1, \beta_2, \dots, \beta_{m_1})$$

$$Z^{(1)} := (1, z_1, z_2, \dots, z_{m_1})^T, \quad W^{(1)} := \begin{pmatrix} w_{0,1} & \dots & w_{m,1} \\ \vdots & \ddots & \vdots \\ w_{0,m_1} & \dots & w_{m,m_1} \end{pmatrix}$$

Hierbei handelt es sich um ein Modell mit einem Hidden Layer, weil wir nur einmal abgeleitete Einflussvariablen gebildet haben. Man kann den Prozess der Bildung abgeleiteter Einflussvariablen aber beliebig oft wiederholen. Dann hat man ein Modell mit mehreren Hidden Layer. Hierzu nimmt man die abgeleiteten Einflussvariablen und bildet aus diesen nochmal neue abgeleitete Einflussvariablen, in dem man sie mit neuen Gewichten versieht und in die Aktivierungsfunktion steckt.

Modell mit 1 Hidden Layer:

$$y = g(\beta Z^{(1)}) + \varepsilon = g(\beta \sigma(W^{(1)} X)) + \varepsilon$$

Modell mit 2 Hidden Layer:

$$Z^{(1)} := \sigma(W^{(1)}X), \quad Z^{(2)} := \sigma(W^{(2)}Z^{(1)})$$

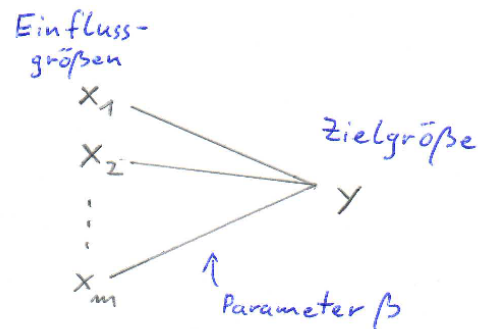
$$y = g(\beta Z^{(2)}) + \varepsilon = g\left(\beta \sigma(W^{(2)} \sigma(W^{(1)}X))\right) + \varepsilon$$

Modell mit k Hidden Layer:

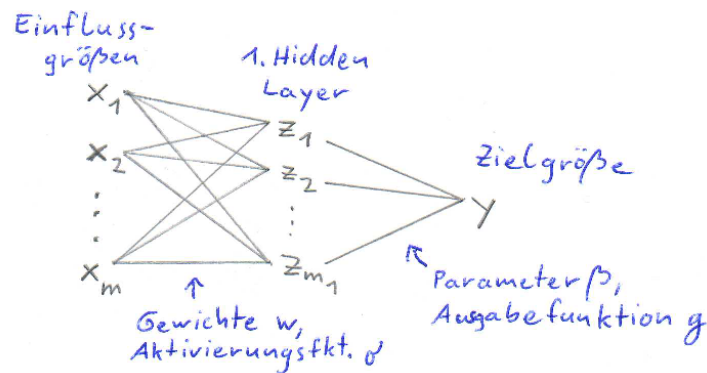
$$Z^{(1)} := \sigma(W^{(1)}X), \quad Z^{(2)} := \sigma(W^{(2)}Z^{(1)}), \quad \dots, \quad Z^{(k)} := \sigma(W^{(k)}Z^{(k-1)})$$

$$\begin{aligned} y &= g(\beta Z^{(k)}) + \varepsilon = \\ &= g\left(\beta \sigma(W^{(k)} \sigma(\dots \sigma(W^{(2)} \sigma(W^{(1)}X)) \dots))\right) + \varepsilon \end{aligned}$$

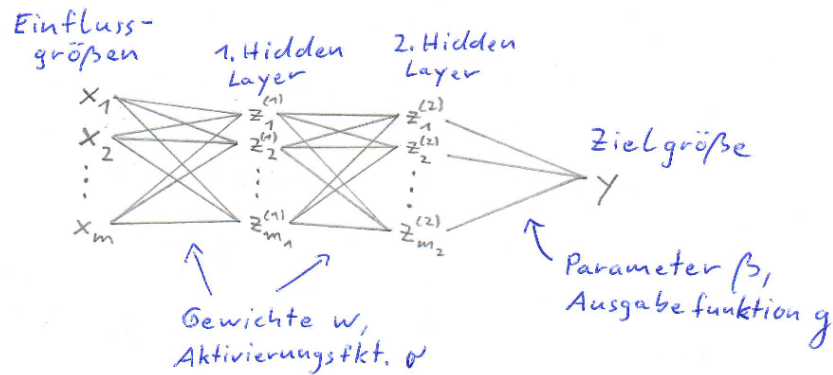
Neuronale Netze lassen sich auch grafisch anschaulich darstellen.
Zunächst grafisch das Modell im Fall der linearen Regression:



Neuronales Netz mit 1 Hidden Layer

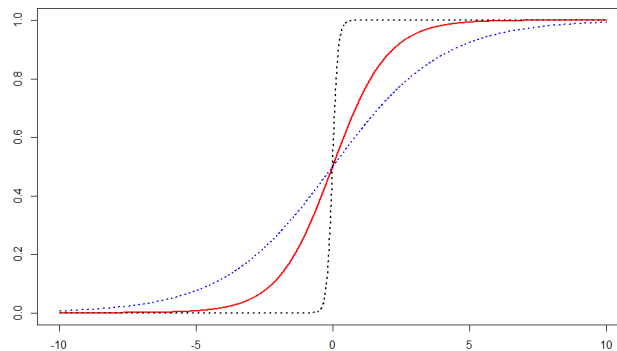


Neuronales Netz mit 2 Hidden Layer



Typische Aktivierungsfunktionen σ :

Sigmoid-Funktion mit Skalenparameter s : $\sigma(t) = \frac{1}{1+\exp(-st)}$



Sigmoid-Funktion mit $s = 1$ (rot), $s = 0,5$ (blau) und $s = 10$ (schwarz)

Typische Ausgabefunktionen g :

- Regression: identische Funktion $g(t) = t$
- Klassifikation: softmax Funktion

Berechnung des Neuronalen Netzes mit Backpropagation:

Um ein Neuronales Netz zu trainieren, das dann gute Prognosen liefern soll, müssen die Gewichte $w_{j,l}$ und die Parameter β_j gut gewählt werden. Hierzu kann zum Beispiel der Backpropagation-Algorithmus verwendet werden.

Diesen Algorithmus sehen wir uns im Folgenden am Beispiel der Regression mit $g(t) = t$ bei nur einem Hidden Layer an. Der allgemeine Fall funktioniert im Wesentlichen identisch, ist aber formelmäßig aufwändiger.

Wir verwenden wieder unsere kompakte Schreibweise:

$$X := (1, x_1, x_2, \dots, x_m)^\top, \quad \beta := (\beta_0, \beta_1, \beta_2, \dots, \beta_{m_1})$$

$$Z^{(1)} := (1, z_1, z_2, \dots, z_{m_1})^\top, \quad W^{(1)} := \begin{pmatrix} w_{0,1} & \dots & w_{m,1} \\ \vdots & \ddots & \vdots \\ w_{0,m_1} & \dots & w_{m,m_1} \end{pmatrix}$$

Im Fall der linearen Regression haben wir also

$$Y = \beta X + \varepsilon$$

Die Regressionskoeffizienten β werden in der linearen Regression durch die Lösung des Optimierungsproblems

$$\min_{\beta} \sum_{i=1}^n (y_i - \beta X_i)^2$$

berechnet. Hierbei handelt es sich um ein relativ einfaches konvexes Optimierungsproblem, dass analytisch mit Hilfe von Matrizenrechnungen gelöst werden kann.

Im Fall des Neuronalen Netzes mit 1 Hidden Layer haben wir

$$Y = g(\beta \sigma(W^{(1)} X)) + \varepsilon$$

und das entsprechende Optimierungsproblem lautet

$$\min_{\beta, W^{(1)}} \sum_{i=1}^n \left(y_i - g(\beta \sigma(W^{(1)} X_i)) \right)^2$$

Hierbei handelt es sich um ein schwieriges und nicht-konvexes Optimierungsproblem, das näherungsweise mit Hilfe eines Optimierungsalgorithmus gelöst werden muss.

Hierzu wird häufig **Backpropagation** verwendet. Dabei handelt es sich um ein Gradientenverfahren. Die Grundidee eines jeden Gradientenverfahrens wird durch folgendes Bild gut dargestellt:



Der Gradient ist die Ableitung einer Funktion mit mehreren Variablen. Dabei ist der Gradient ein Wegweiser, der einem den Weg “nach unten” zeigt. Das Verfahren beginnt an einem irgendwie gewählten Startpunkt und wandert dann schrittweise über die zu optimierende Funktion. Bei jedem Schritt wird der Gradient berechnet und als Wegweiser für den nächsten Schritt verwendet. Das Verfahren endet, wenn es in einer Talsohle angekommen ist – und es somit in keiner Richtung mehr weiter nach unten geht.

Gradientenverfahren gibt es in verschiedenen Varianten. Um die Grundidee des Gradientenverfahrens “Backpropagation” besser zu verstehen, beschränken wir uns im folgenden auf den Spezialfall

- Regression: Ausgabefunktion $g(t) = t$
- Neuronales Netz mit 1 Hidden Layer

Die Grundidee ist im allgemeinen Fall identisch, aber die Formeln sind unübersichtlicher. Für den gewählten Spezialfall lautet das Optimierungsproblem:

$$\min_{\beta, W} \sum_{i=1}^n \underbrace{(y_i - \beta_0 - Wx_i)^2}_{= f(x_i)} \\ \underbrace{\hspace{10em}}_{=: R_i(\beta, W)}$$

Hieraus ergeben sich die Gradienten (“Wegweiser zum Optimum”) durch partielles Ableiten :

$$\frac{\partial R_i}{\partial \beta_j} = \underbrace{-2(y_i - f(x_i))}_{=: \delta_i} z_{j,i}$$

$$\frac{\partial R_i}{\partial w_{j,l}} = \underbrace{-2(y_i - f(x_i))\beta_j \sigma'(w_l^T x_i)}_{=: s_{j,i}} x_{l,i}$$

Setzt man die obere Formel in die untere Formel ein, dann erhält man die Backpropagation-Gleichung:

$$s_{j,i} = \sigma'(w_l^T x_i) \beta_j \delta_i$$

Der Name “Backpropagation” kommt daher, weil man sich bei der Berechnung dieser Gleichung rückwärts durch das neuronale Netz bewegt. Nach der Berechnung des Gradienten erfolgt der Update-Schritt:

$$\beta_j^{\text{neu}} = \beta_j^{\text{alt}} - \gamma \sum_{i=1}^n \frac{\partial R_i}{\partial \beta_j} (\beta^{\text{alt}}, w^{\text{alt}})$$

$$w_{j,l}^{\text{neu}} = w_{j,l}^{\text{alt}} - \gamma \sum_{i=1}^n \frac{\partial R_i}{\partial w_{j,l}} (\beta^{\text{alt}}, w^{\text{alt}})$$

Die neuen Werte für β und W ergeben sich, indem zu den alten Werten der (negative) Gradient addiert wird. Der Faktor $\gamma \in (0, \infty)$ ist dabei die sogenannte Schrittweite und gibt an, in welchen Abständen neue Gradienten (“Wegweiser”) berechnet werden.

4.4 k -Nearest Neighbors

Das Verfahren k -Nearest Neighbor ist eines der einfachsten maschinellen Lernverfahren und kann sowohl für Regression als auch für Klassifikation

verwendet werden. Wie der Name des Verfahrens schon sagt, spielen dabei für Prognosen die k nächsten Nachbarn eines Datenpunktes eine entscheidende Rolle. Wenn zum Beispiel $k = 12$ ist, dann errechnet sich die Prognose $\hat{y}_{\text{neu}}$ für die Zielvariable y_{neu} eines neuen Datenpunkts x_{neu} (Einflussvariablen) folgendermaßen: Aus den Daten $x_1, x_2, x_3, \dots, x_n$ des Datensatzes werden die 12 nächsten Datenpunkte zu x_{neu} herausgesucht. Das sind diejenigen Datenpunkte $x_{i_1}, x_{i_2}, \dots, x_{i_{12}}$, die dem Punkt entfernungsmäßig am nächsten liegen (geometrischer Abstand).

Im Fall der **Regression** ist die Prognose $\hat{y}_{\text{neu}}$ für den neuen Datenpunkt x_{neu} dann einfach das arithmetische Mittel (Durchschnitt) über die entsprechenden Werte $y_{i_1}, y_{i_2}, \dots, y_{i_{12}}$ der Zielvariablen dieser $k = 12$ nächsten Nachbarn:

$$\hat{y}_{\text{neu}} = \frac{y_{i_1} + y_{i_2} + \dots + y_{i_k}}{k}$$

Im Fall der **Klassifikation** wird die Prognose $\hat{y}_{\text{neu}}$ für den neuen Datenpunkt x_{neu} über eine einfache Mehrheitsentscheidung getroffen. Die Kategorie der (kategoriellen) Zielvariablen y , die bei den $k = 12$ nächsten Nachbarn am häufigsten vorkommt, wird als Prognose $\hat{y}_{\text{neu}}$ für den neuen Datenpunkt x_{neu} genommen. Die relativen Häufigkeiten stellen dabei auch Prognosen für die Wahrscheinlichkeiten dar, mit der neue Datenpunkt in welche Kategorie der Zielvariable fällt. Ist die Zielvariable y eine 0-1-Variable (binäre Klassifikation), dann ist also

$$\hat{y}_{\text{neu}} = \frac{y_{i_1} + y_{i_2} + \dots + y_{i_k}}{k}$$

die (prognostizierte) Wahrscheinlichkeit dafür, dass y_{neu} den Wert 1 hat.

Bemerkungen

- Bei der Bestimmung der k nächsten Nachbarn werden die Entfernungen normalerweise mit dem normalen euklidischen Abstand berechnet.
- Da das Verfahren auf Abstände zwischen Datenpunkten beruht, ist es eigentlich nur sinnvoll, wenn alle Einflussvariablen metrisch sind. (Bei nominalen und ordinalen Daten gibt es keine sinnvollen Abstände!)
- Wenn Entfernungen durch den normalen euklidischen Abstand berechnet werden, dann spielt für die Größe des Abstands eine Rolle, in welchen Einheiten die Daten gemessen werden.

Beispiel Körpergröße in Meter oder Zentimeter:

Betrachten wir drei Personen: Person 1 hat eine Größe von 1,75 m und ein Gewicht von 69 kg. Person 2 hat eine Größe von 1,70 m und ein Gewicht von 67 kg. Person 3 hat eine Größe von 1,76 m und ein Gewicht von 72 kg. Wir haben also die drei Datenpunkte

$$P_1 = (1,75; 69), \quad P_2 = (1,7; 67) \quad \text{und} \quad P_3 = (1,76; 72).$$

Liegt P_2 oder P_3 näher an P_1 ? Der Abstand zwischen P_1 und P_2 beträgt

$$\text{Abstand}(P_1, P_2) = \sqrt{(1,75 - 1,7)^2 + (69 - 67)^2} \approx 2,0$$

Der Abstand zwischen P_1 und P_3 beträgt

$$\text{Abstand}(P_1, P_3) = \sqrt{(1,75 - 1,76)^2 + (69 - 72)^2} \approx 3,0.$$

Somit liegt P_2 näher an P_1 . Also ist P_2 der nächste Nachbar von P_1 . Wandeln wir aber die Körpergröße in die Einheit cm um, dann haben wir die drei Punkte

$$P_1 = (175; 69), \quad P_2 = (170; 67) \quad \text{und} \quad P_3 = (176; 72).$$

Die entsprechenden Abstände sind nun

$$\text{Abstand}(P_1, P_2) = \sqrt{(175 - 170)^2 + (69 - 67)^2} \approx 5,4$$

und

$$\text{Abstand}(P_1, P_3) = \sqrt{(175 - 176)^2 + (69 - 72)^2} \approx 3,2.$$

Im Gegensatz zu vorher liegt nun P_3 näher an P_1 , so dass nun P_3 der nächste Nachbar von P_1 ist. Das heißt es hängt von der willkürlich gewählten Einheit bei der Dateneingabe ab, welche Punkte die nächsten Nachbarn sind.

Damit es deswegen nicht zu willkürlichen und verzerrten Ergebnissen kommt, ist es daher sinnvoll die Daten der Einflussvariablen zu standardisieren. Dazu wird bei den Einflussvariablen im Datensatz in jeder Spalte jeweils der Spaltenmittelwert abgezogen und durch die Standardabweichung (der Daten aus der entsprechenden Spalte) geteilt

$$\tilde{x}_i = \frac{x_i - \text{Mittelwert}}{\text{Standardabweichung}}$$

- Bei der Regression beruht das Verfahren auf dem arithmetischen Mittel. Das Verfahren ist daher empfindlich gegenüber Ausreißern. Alternativ kann statt dem arithmetischen Mittel auch der Median (unempfindlich gegenüber Ausreißern) verwendet werden.

Entsprechend kann auch beim Standardisieren der Daten der Mittelwert durch den Median und die Standardabweichung durch den Interquartilsabstand ersetzt werden.

- Die Anzahl k der nächsten Nachbarn muss – abhängig von den Daten – gut gewählt werden. Es handelt sich dabei um einen Tuningparameter, der die Komplexität der resultierenden Prognosefunktion steuert. Details hierzu später.

5 Bewertung von Prognosen: Gütemaße im Supervised Learning

5.1 Einführung zu Gütemaße

Mean Squared Error (MSE)

Sei $f : \mathbb{R}^m \rightarrow \mathbb{R}$ eine Prognosefunktion, die zu jedem Datenpunkt x_i (Einflussvariablen) eine Prognose $\hat{y}_i = f(x_i)$ für die Zielvariable y_i berechnet. Je näher die Prognose $\hat{y}_i = f(x_i)$ am tatsächlichen Wert y_i der Zielvariable liegt, desto besser ist die Prognose. Das populärste Gütemaß zur Beurteilung einer Prognosefunktion ist der *Mean Squared Error* (MSE):

$$\text{MSE}(f) = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2$$

Hierbei handelt es sich also – wie der Name schon sagt – um den durchschnittlichen quadratischen Fehler. Die Popularität des MSE hat eher historische mathematische Gründe statt inhaltliche Gründe. Inhaltlich ist ein anderes Gütemaß oft sinnvoller wie man am Beispiel der Nachfrageprognose sieht.

Beispiel Nachfrageprognose

Stellen Sie sich vor, Sie haben einen Algorithmus geschrieben, der basierend auf irgendwelchen Einflussvariablen x die Nachfrage y nach Äpfeln prognostiziert. Dieser Algorithmus stellt also eine Prognosefunktion $f : \mathbb{R}^m \rightarrow \mathbb{R}$ für die Nachfrage dar. Um die Qualität der Prognosen Ihres Algorithmus zu bewerten, wäre zum Beispiel interessant, um wieviele Äpfel sich der Algorithmus im Durchschnitt verschätzt:

$$\text{durchschnittlicher Prognosefehler des Algorithmus} = \frac{1}{n} \sum_{i=1}^n |f(x_i) - y_i|$$

Dies ist die sogenannte *Mean Absolute Error* (MAE):

$$\text{MAE}(f) = \frac{1}{n} \sum_{i=1}^n |f(x_i) - y_i|.$$

MSE und MAE unterscheiden sich nur durch das Quadrat (beim MSE). Während die inhaltliche Interpretation des MSE aber (wegen dem Quadrat) ziemlich unklar ist, hat der MAE eine klare und einfache Bedeutung: $\text{MAE}(f) = 6,2$ bedeutet, dass der Algorithmus die tatsächliche Nachfrage durchschnittlich bis auf $\pm 6,2$ Äpfel genau schätzt.

Was ist besser: Mittelwert oder Median?

Angenommen Sie sollen nun die Nachfrage nach Äpfeln für die nächste Woche prognostizieren, und zwar basierend auf der Nachfrage nach Äpfeln der letzten 100 Wochen: $y_1, \dots, y_{100}$. Die naheliegendste Prognose ist sicherlich die durchschnittliche Nachfrage der letzten 100 Wochen

$$\hat{y}_{\text{mean}} = \frac{y_1 + \dots + y_{100}}{100}.$$

Eine Alternative wäre aber auch der Median

$$\hat{y}_{\text{median}} = \text{Median}(y_1, \dots, y_{100}).$$

Was liefert nun die besseren Prognosen, der Mittelwert $\hat{y}_{\text{mean}}$ oder der Median $\hat{y}_{\text{median}}$?

Die Antwort dieser Frage hängt vom gewählten Gütemaß ab. Wählt man den MSE als Gütemaß, dann ist der Mittelwert die bestmögliche Prognose, wählt man jedoch den (inhaltlich sinnvolleren) MAE als Gütemaß, dann ist der Median die bestmögliche Prognose.

An dieser Stelle sollten Sie sich unbedingt schon einmal zwei Dinge merken:

1. Welcher Prognosealgorithmus “besser” ist, hängt vom gewählten Gütemaß ab und für das Gütemaß gibt es immer verschiedene Alternativen. Bei der Konstruktion eines Prognosealgorithmus sollte man sich also vorher Gedanken machen, welches Gütemaß am sinnvollsten gewählt werden sollte!
2. Der Median (und nicht der Mittelwert) minimiert den mittleren Prognosefehler. Will man also den mittleren Prognosefehler minimieren, dann sollte man nicht den Mittelwert, sondern den Median verwenden!

Wie man mit Prognosen verhindert, dass die Äpfel ausgehen

Lagerhaltung mit dem Median:

Wie wir gesehen haben, liefert der Median hier “bessere” Prognosen als der Mittelwert. Einfach nur den Median zu verwenden wäre aber für den Äpfelverkäufer ruinös. Angenommen die durch den Median errechnete Prognose für die Äpfelnachfrage wäre 325. Die tatsächliche Nachfrage wird aber sicherlich etwas höher oder etwas niedriger sein, da die Prognose ja nicht 100-prozentig genau eintreffen wird. Hätte der Äpfelverkäufer genau 325 Äpfel auf Lager, dann wäre die Wahrscheinlichkeit, dass er zu wenige Äpfel hat, gleich 50%. (Die 50% kommen daher, da 325 in dem Beispiel genau der Median ist.) Das heißt, nur mit 50% Wahrscheinlichkeit würden die Lagerbestände bei diesem Vorgehen ausreichen.

Alternative 1: Mittelwert und Standardabweichung

Der klassische Weg dieses Problem zu lösen, verwendet den Mittelwert und die Standardabweichung. Hierzu wird aus den vergangenen Nachfragewerten $y_1, \dots, y_n$ der Mittelwert $\hat{y}_{\text{mean}}$ und die Standardabweichung $\hat{\sigma}$ berechnet. Hat man dann

$$\hat{y}_{\text{mean}} + 2\hat{\sigma}$$

Äpfel auf Lager, dann reichen die Lagerbestände mit einer Wahrscheinlichkeit von 97,5% – vorausgesetzt, dass die Nachfrage nach Äpfeln normalverteilt ist. Häufig sind Nachfrage und Bedarfe aber überhaupt nicht normalverteilt (sondern deutlich schief-verteilt). In diesem Fall führt die obige klassische Formel zu falschen Werten. Erstaunlicherweise verwenden auch heutige Lehrbücher zum Bestandsmanagement immer noch diese Formel.

Alternative 2: Quantile

Die naheliegende Lösung des Problems besteht in der Verwendung von Quantilen. Hierzu berechnet man aus den vergangenen Nachfragewerten $y_1, \dots, y_n$ das 97,5%-Quantil $\hat{q}_{0.975}$. Hat man $\hat{q}_{0.975}$ Äpfel auf Lager, dann reicht der Bestand – wie sich einfach aus der Definition des Quantils ergibt – mit einer Wahrscheinlichkeit von 97,5%.

5.2 Gütemaße in Regressionsproblemen

In einer Regression wird eine Regressionsfunktion $f : \mathbb{R}^m \rightarrow \mathbb{R}$ berechnet. Da man mit Hilfe dieser Funktion für Datenpunkte x (Einflussvariablen) Prognosen $\hat{y} = f(x)$ für die Zielvariable y berechnen kann, nennt man die

Funktion f auch Prognosefunktion. In der linearen Regression hat die Regressionsfunktion/Prognosefunktion die Form

$$f(x_i) = f(x_{i,1}, \dots, x_{i,m}) = \beta_0 + \beta_1 x_{i,1} + \dots + \beta_m x_{i,m}$$

für $x_i = (x_{i,1}, \dots, x_{i,m}) \in \mathbb{R}^m$, wobei $\beta_0, \beta_1, \dots, \beta_m$ die Regressionskoeffizienten sind. (Erinnern Sie sich hierzu daran, wie wir die Prognosen bei der linearen Regression berechnet haben.)

Haben wir in einer Regression m Einflussvariablen, dann besteht eine Zeile aus den Werten $x_{i,1}, \dots, x_{i,m}$ für die Einflussvariablen und y_i für die Zielvariable. Wir haben also $m + 1$ Spalten und i ist die Nummer der Zeile. Die meisten Gütemaße für die Regression haben dann die Form

$$\text{Gütemaß}(f) = \sum_{i=1}^n \frac{L(f(x_{i,1}, \dots, x_{i,m}), y_i)}{n},$$

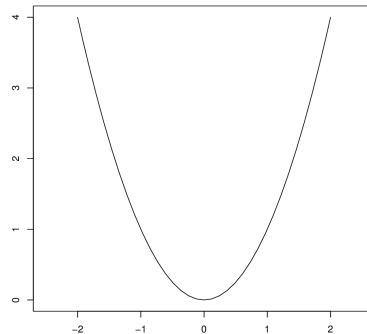
wobei $L : \mathbb{R}^2 \rightarrow \mathbb{R}$ eine sogenannte Verlustfunktion ist. Wir haben im Einleitungskapitel schon zwei Beispiele hierfür kennengelernt. Beim MSE ist die Verlustfunktion die quadratische Funktion, beim MAE ist die Verlustfunktion die Betragsfunktion und bei Quantilen ist die sogenannte Pinball-Verlustfunktion die zu verwendende Verlustfunktion:

- MSE für Mittelwertregression:

$$\text{MSE}(f) = \sum_{i=1}^n \frac{L(f(x_{i,1}, \dots, x_{i,m}), y_i)}{n}$$

mit der quadratischen Verlustfunktion

$$L(\hat{y}, y) = (\hat{y} - y)^2.$$

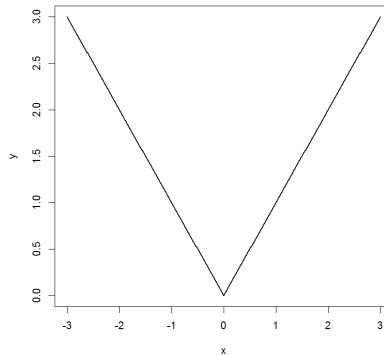


- MAE für Medianregression:

$$\text{MAE}(f) = \sum_{i=1}^n \frac{L(f(x_{i,1}, \dots, x_{i,m}), y_i)}{n}$$

mit der Betrags-Verlustfunktion

$$L(\hat{y}, y) = |\hat{y} - y|.$$



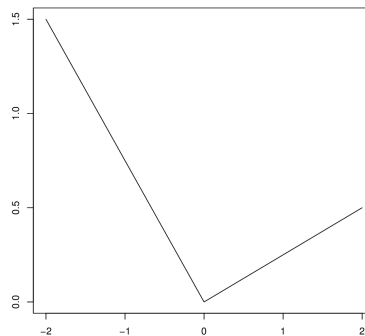
- Für die Quantilsregression:

$$\text{Gütemaß}(f) = \sum_{i=1}^n \frac{L_q(f(x_{i,1}, \dots, x_{i,m}), y_i)}{n}$$

mit der Pinball-Verlustfunktion

$$L_q(\hat{y}, y) = (1-q) \cdot |\hat{y} - y| \text{ für } \hat{y} < y \quad \text{und} \quad L_q(\hat{y}, y) = q \cdot |\hat{y} - y| \text{ für } \hat{y} \geq y,$$

wobei q das gewählte Quantil ist.



Eine weiteres populäres Gütemaß ist der MAPE:

- MAPE:

$$\text{MAPE}(f) = \sum_{i=1}^n \frac{L(f(x_{i,1}, \dots, x_{i,m}), y_i)}{n}$$

mit der Verlustfunktion

$$L(\hat{y}, y) = \frac{|\hat{y} - y|}{|y|}.$$

MAPE steht für Mean Absolute Percentage Error. Es handelt sich dabei also um den durchschnittlichen prozentualen Prognosefehler. Der Gedanke hinter dem Mape ist, den Prognosefehler ins Verhältnis zum absoluten Wert zu setzten. Das klingt einleuchtend, hat aber einen schweren Nachteil, v.a. wenn die Werte der Zielvariablen y stark schwanken. In diesem Fall ist der MAPE dann eher irreführend und wenig aussagekräftig. Betrachten Sie hierzu folgende Beispielaufgabe.

Aufgabe: Stellen Sie sich vor, sie haben einen Prognosealgorithmus entwickelt, der die Nachfrage nach Äpfeln durchschnittlich bis auf einen Apfel genau prognostizieren kann. Der Algorithmus wird nun an zwei Tagen eingesetzt. Am ersten Tag prognostiziert Ihr Algorithmus eine Nachfrage von 101 Äpfel, aber es werden genau 100 Äpfel nachgefragt. Am zweiten Tag prognostiziert Ihr Algorithmus eine Nachfrage von 2 Äpfeln, aber es wird nur 1 Apfel nachgefragt. Berechnen Sie nun den MSE, den MAE und den MAPE. Was denken Sie?

5.3 Gütemaße in Klassifikationsproblemen

In Klassifikationsproblemen können Gütemaße ebenfalls wie in Regressionsproblemen über eine Verlustfunktion definiert werden. Das einfachste und wichtigste Beispiel hierfür ist die Misclassification Error Rate.

- Misclassification Error Rate für Klassifikation:

$$\text{MisclassificationErrorRate}(f) = \sum_{i=1}^n \frac{L(f(x_{i,1}, \dots, x_{i,m}), y_i)}{n}$$

mit der Misclassification-Verlustfunktion

$$L(\hat{y}, y) = 1 \text{ für } \hat{y} \neq y \quad \text{und} \quad L(\hat{y}, y) = 0 \text{ für } \hat{y} = y.$$

Bei einer (binären) Klassifikation gibt es nur zwei Möglichkeiten, nämlich $y = 0$ oder $y = 1$. Die Prognose $\hat{y} = 0$ oder $\hat{y} = 1$ ist also entweder richtig oder falsch. (Nur ein bisschen falsch wie in der Regression geht hier nicht.) Die Misclassification Error Rate zählt also lediglich, wie oft die Prognose falsch war (und errechnet daraus den prozentualen Anteil an falschen Prognosen).

Die Verwendung der Misclassification Error Rate ist naheliegend, aber in vielen Fällen nicht aussagekräftig. Vor allem in folgenden Fällen sollte die Misclassification Error Rate nicht verwendet werden:

- Die Klassen $y = 0$ und $y = 1$ sind nicht annähernd gleich groß:

Stellen Sie sich vor, Sie sollen einen Test entwickeln, der bei Personen eine bestimmte Krankheit erkennen soll. Die Krankheit ist aber glücklicherweise relativ selten, im Durchschnitt hat nur 1 von 10000 Personen die Krankheit. Wenn Ihr Testverfahren anhand der Misclassification Error Rate bewertet wird, können Sie sich entspannt zurücklehnen. Schreiben Sie einfach einen Algorithmus, der bei jeder Person immer "gesund" prognostiziert. Damit erreichen Sie – einfach weil die Krankheit relativ selten ist – eine fantastische Misclassification Error Rate von nur 0,0001. Sagenhafte 99,99% Ihrer Prognosen sind korrekt!

- Fehler wiegen unterschiedlich schwer:

Vor allem bei medizinischen Tests wiegen Fehler unterschiedlich schwer. Hierbei gibt es 2 unterschiedliche Fehler:

1. Nicht entdeckte Krankheit: Obwohl die Person krank ist ($y = 1$), wird die Krankheit nicht entdeckt ($\hat{y} = 0$).
2. Falscher Alarm: Obwohl die Person nicht krank ist ($y = 0$), entscheidet der Test fälschlicherweise auf "krank" ($\hat{y} = 1$).

Der 1. Fehler (nicht entdeckte Krankheit) kann schwerwiegende Folgen haben. Wenn dadurch z.B. eine Krebserkrankung zu spät festgestellt wird, kann dadurch die Überlebenschance des Patienten dramatisch sinken. Der 2. Fehler (falscher Alarm) wird demgegenüber häufig als nicht so schwerwiegend eingeschätzt (weil er meist weniger dramatische Folgen hat und bei Nachfolgeuntersuchungen oft schnell erkannt wird).

Da in vielen Fällen die Misclassification Error Rate wenig aussagekräftig ist, sollte man sich bei Klassifikationen die komplette Vierfelder-Tafel anschauen. Diese kann zum Beispiel so aussehen:

	Realität		
	$y = 0$	$y = 1$	
Prognose $\hat{y} = 0$	317	6	323
$\hat{y} = 1$	93	28	121
	410	34	444

Das heißt also, im Beispiel mit dem Krankheitstest würde 6 mal die Krankheit bei erkrankten Personen nicht erkannt und es gäbe 93 Fehlalarme. Insgesamt wären 34 Personen tatsächlich krank und 444 Personen wären in dem Zahlenbeispiel untersucht worden.

Aus den vier Zahlen im Zentrum der Vierfelder-Tafel (d.h. 317, 6, 93 und 28) lassen sich eine Unmenge von Kennzahlen errechnen, die auch alle einen eigenen Namen haben. Es ist aber müßig, diese Liste an Kennzahlen mit Namen auswendig zu lernen. Stattdessen sollte man in einem Anwendungsfall stets überlegen, welche Zahl(en) von größter Bedeutung sind und diese dann errechnen. Also selber denken statt auswendig lernen! Hierzu ein Beispiel:

Beispiel:

In einem meiner Projekte sollte ein System entwickelt werden, dass – basierend auf maschinellern Lernen – Angriffe auf IT-Systeme erkennt. Es gibt also zwei Möglichkeiten für die Realität: es ist ein Angriff erfolgt ($y = 1$) oder es ist kein Angriff erfolgt ($y = 0$). Ebenso gibt es für das Warnsystem zwei Möglichkeiten: es wird ein Alarm ausgegeben ($\hat{y} = 1$), oder es wird kein

Alarm ausgegeben ($\hat{y} = 0$).

In diesem Beispiel ist sicherlich der Anteil an nicht entdeckten Angriffe von zentraler Bedeutung - das wäre in obigem Zahlenbeispiel also

$$\text{Anteil nicht entdeckter Angriffe} = \frac{6}{6 + 28} \approx 17,6\%.$$

Versetzt man sich in die Lage der menschlichen Kontrollperson, dann wird klar, dass der Anteil an Fehlalarmen unter allen Alarmen ebenfalls von großer Bedeutung ist. Nehmen wir an, der Anteil an Fehlalarmen unter allen Alarmen wäre mit 99,8 % sehr hoch. Das würde bedeuten, dass von 1000 Alarmen, die das System ausgibt, nur 2 Alarme berechtigt wären und 998 Alarme wären Fehlalarme. Das ist schlecht, weil dann die Gefahr besteht, dass die menschliche Kontrollperson Alarme irgendwann nicht mehr ernst nimmt und einfach ungeprüft wegdrückt ("Ist ja eh wieder nur ein Fehlalarm."). In obigem Zahlenbeispiel wäre diese Kennzahl also

$$\text{Anteil der Fehlalarme unter allen Alarmen} = \frac{93}{93 + 28} \approx 76,9\%.$$

Anstatt Listen mit 100 000 Kennzahlen auswendig zu lernen und im Anwendungsfall zu berechnen, ist hier also "Denken von Fall zu Fall" sehr viel sinnvoller!

Es gibt jedoch zwei Kennzahlen, die in der Medizin üblicherweise verwendet werden, und die man schon aus Allgemeinbildungsgründen kennen sollte: Sensitivität und Spezifität. Diese beiden Kennzahlen schauen wir uns kurz an, dass heißt aber nicht, dass diese Kennzahlen immer von besonderer Bedeutung wären und standardmäßig berechnet werden sollten.

- Sensitivität: Die Sensitivität ist die Empfindlichkeit oder auch Richtig-positiv-Rate eines Klassifikators. Im Beispiel mit der Diagnose einer Krankheit, wäre das der Anteil der richtig identifizierten kranken Personen ($\hat{y} = 1$ und $y = 1$) unter allen kranken Personen ($y = 1$). Im obigen Zahlenbeispiel wäre das

$$\text{Sensitivität} = \frac{\hat{y} = 1 \text{ und } y = 1}{y = 1} = \frac{28}{6 + 28} \approx 82,4\%.$$

- Spezifität: Die ist die Richtig-negativ-Rate eines Klassifikators. Im Beispiel mit der Diagnose einer Krankheit, wäre das der Anteil der richtig

identifizierten gesunden Personen ($\hat{y} = 0$ und $y = 0$) unter allen gesunden Personen ($y = 0$). Im obigen Zahlenbeispiel wäre das

$$\text{Spezifität} = \frac{\hat{y} = 0 \text{ und } y = 0}{y = 0} = \frac{317}{317 + 93} \approx 77,3\%.$$

5.4 Trainings- und Testdatensatz

Im vorherigen Unterkapitel wurden Kennzahlen vorgestellt, mit denen die Genauigkeit von Prognosen bewertet werden kann. Jedes maschinelle Lernverfahren im Supervised Learning berechnet ein Modell für den Zusammenhang zwischen den Einflussvariablen und der Zielvariable. Dieser Zusammenhang wird dargestellt durch eine Funktion

$$f : \mathbb{R}^m \rightarrow \mathbb{R},$$

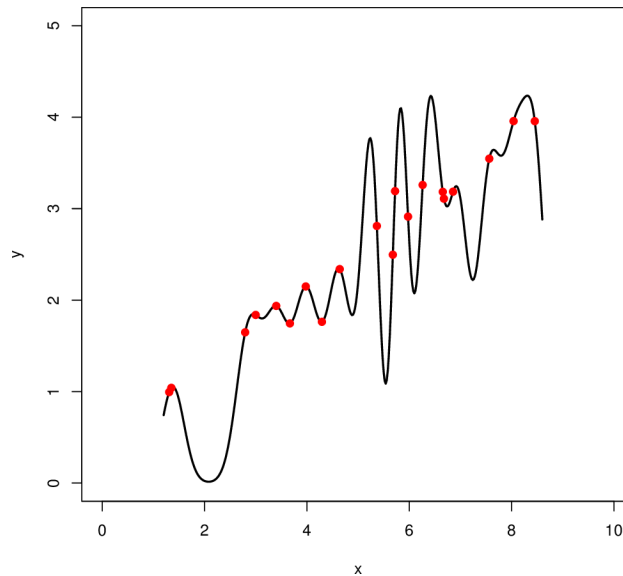
wobei m die Anzahl der Einflussvariablen ist. Diese Funktion $f : \mathbb{R}^m \rightarrow \mathbb{R}$ heißt auch Prognosefunktion, weil mit ihr zu jedem Datenpunkt x (Einflussvariablen) eine Prognose $\hat{y} = f(x)$ für die Zielvariable y berechnet wird.

Aus der Genauigkeit dieser Prognosen folgt, wie gut das Modell f die Realität abbildet: je genauer die Prognosen, desto besser das Modell. Um die Genauigkeit von Prognosen zu errechnen, haben wir schon die sogenannte *Mean Absolute Error* (MAE) kennengelernt:

$$\text{MAE}(f) = \frac{1}{n} \sum_{i=1}^n |y_i - f(x_i)|.$$

Der MAE ist die mittlere Abweichung der Prognosen $f(x_i)$ von der Realität y_i .

Wird aber nun die Prognosegenauigkeit anhand derselben Daten berechnet, mit denen die Prognosefunktion f berechnet wurde, dann ergeben sich Probleme. Betrachten wir dazu folgendes Beispiel mit Daten und einer Modellfunktion f :



Obwohl das ein sehr schlechtes Modell ist, hat diese Modellfunktion f aber einen Prognosefehler von 0 auf dem Datensatz und hätte somit einen optimalen Fehlerwert $\text{MAE}(f) = 0$. Um die Genauigkeit der Prognosen und damit das Modell f zu bewerten, darf man nicht den selben Datensatz verwenden, mit dem das Modell (also die Funktion f) schon berechnet wurde. Denn bei der Berechnung von f hat der Algorithmus die Daten ja bereits gesehen und berücksichtigt. Das wäre dann in etwa so, wie wenn man nicht die Zukunft, sondern die Vergangenheit vorhersagen müsste.

Für eine valide Bewertung der Prognosegenauigkeit braucht man daher zwei getrennte Datensätze:

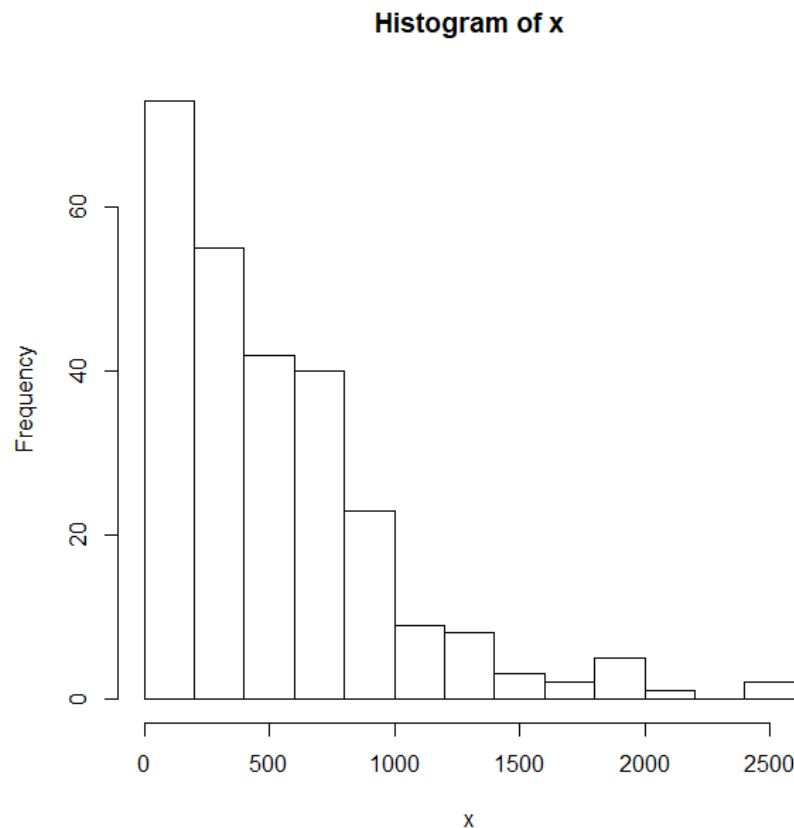
1. Trainingsdatensatz: Der Trainingsdatensatz wird verwendet, um das Modell f zu berechnen, also das neuronale Netz, den Entscheidungsbaum, etc.
2. Testdatensatz: Auf dem Testdatensatz wird dann die Genauigkeit der Prognosen berechnet, die das Modell f erzeugt. Das heißt der MAE oder die Misclassification Error wird auf dem Testdatensatz berechnet.

In der Praxis wird das dadurch umgesetzt, dass die verfügbaren Daten zufällig in zwei Teile geteilt werden. Die Aufteilung ist meistens 70% zu 30%. Der größere Teildatensatz (70%) wird als Trainingsdatensatz verwendet, der kleinere Teildatensatz (30%) wird als Testdatensatz verwendet. Oftmals sind die

Daten im Datensatz in irgendeiner Weise sortiert. Daher ist es in der Regel wichtig, dass die Daten durch einen Zufallsgenerator aufgeteilt werden.

5.5 Spezial: Schiefe Verteilung in der Zielvariable

Eine spezielle Situation ergibt sich, wenn die Verteilung der Zielvariablen schief ist. Folgendes Histogramm zeigt die Spielergehälter von Baseball-Spielern in den USA:



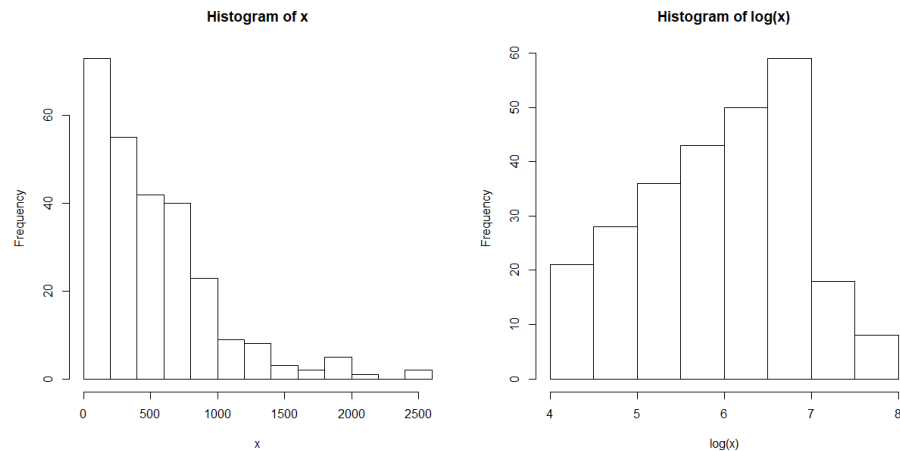
Hierbei handelt es sich um eine extrem schiefe Verteilung. Maschinelle Lernverfahren haben (häufig) Probleme, wenn die Verteilung der Zielvariable schief ist. Dieses Problem tritt auch in der binären Klassifikation auf. In diesem Fall bedeutet eine schiefe Verteilung, dass die Klassen der Zielvariable sehr ungleich verteilt sind (z.B. 90% der Daten in Klasse 0 und nur 10%

der Daten in Klasse 1). Dies bezeichnet man als Imbalanced. In jedem Fall benötigen schiefe Verteilungen in der Zielvariablen ein besonderes Augenmerk. Dabei unterscheidet sich das Vorgehen sehr, je nachdem ob es sich um ein Regressionsproblem oder ein Klassifikationsproblem handelt.

Regression: Schiefe Verteilung in der Zielvariablen

Sehr viele maschinelle Lernverfahren sind Mittelwertverfahren. Mittelwertverfahren versuchen jeweils – in Abhängigkeit der Einflussvariablen – den Mittelwert der Zielvariablen zu schätzen. Hierunter fallen zum Beispiel die lineare Regression, Entscheidungsbäume, Random Forest, LASSO, LS-Support Vector Regression, k -Nearest-Neighbors, L_2 -Boosting und auch Neuronale Netze (je nach verwendeter Verlustfunktion). Wie wir bereits aus der explorativen Datenanalyse wissen, sind Mittelwertberechnungen nicht für schiefe Verteilungen geeignet. Bei einer deutlich schief verteilten Zielvariablen sind also alle soeben genannten maschinellen Lernverfahren nicht ohne Weiteres einsetzbar. In diesem Fall kommen vor allem zwei alternative Vorgehensweisen zum Einsatz:

1. Anstelle von Mittelwertverfahren können Medianverfahren verwendet werden. Beispiele für Medianverfahren sind Support Vector Regression oder Neuronale Netze mit Betragsverlustfunktion (absolute loss). Ob ein Verfahren auf dem Mittelwert oder dem Median beruht, erkennt man in der Regel an der verwendeten Verlustfunktion:
 - quadratische Verlustfunktion = Mittelwertverfahren
 - Betragsverlustfunktion = Medianverfahren.
2. Die Werte der Zielvariablen werden mit einer Transformationsfunktion so transformiert, dass die Werte nicht mehr schief verteilt sind. Grundsätzlich kommt als Transformationsfunktion jede streng monotone Funktion in Frage. in den allermeisten Fällen wird jedoch der natürliche Logarithmus verwendet. Folgende Grafik zeigt links nochmal das Histogramm für die Gehälter der Baseball-Spieler und rechts das Histogramm für die logarithmierten Werte:



Man sieht, dass die Verteilung der logarithmierten Werte zwar nicht perfekt symmetrisch ist, aber die Schiefe der Verteilung konnte so drastisch reduziert werden, dass nun auch Mittelwertverfahren verwendet werden können. Dabei werden die überwachten Lernalgorithmen ganz normal angewendet, nur dass als Zielvariable nun die logarithmierten Werte verwendet werden. Allein bei der Berechnung der Prognosegüte gibt es einen Unterschied: Die Prognosen aus dem Maschinellen Lernverfahren sollten zunächst mit der Exponentialfunktion (= Umkehrfunktion des Logarithmus) rücktransformiert werden, so dass die Prognosegüte (zur besseren Interpretierbarkeit) auf den nicht-transformierten Werten berechnet werden kann.

Bei beiden Vorgehensweisen gilt: Ist die Zielvariable schief, dann ist die Verwendung des MSE als Prognosegüte in der Regel natürlich gänzlich unangebracht; stattdessen sollte der MAE (oder evtl. auch der MAPE) verwendet werden!

Klassifikation: Klassen der Zielvariablen imbalanced

Maschinelle Lernverfahren haben ein Problem, wenn die Klassen der Zielvariablen stark ungleich verteilt sind. In diesem Fall neigen Maschinelle Lernverfahren dazu, der Minderheitenklasse zu wenig und der Mehrheitsklasse zu viel Gewicht beizumessen.

Warum ist das so? Nehmen wir an, dass in der Zielvariablen 90 % der Werte eine 0 sind und nur 10 % der Werte eine 1 sind. Für das Maschinelle Lernverfahren ist daher die Versuchung sehr groß, einfach immer nur 0 zu prognostizieren. Das heißt, die aus den Daten erlernte Prognosefunktion wäre konstant $\hat{f} \equiv 0$. Schließlich liegt die Accuracy dieser Funktion bei sehr guten 90%. (Wobei natürlich – wie Sie bereits bestens wissen – die Accuracy hier völlig nichtssagend ist.)

Es gibt nun zwei alternative Strategien, die beide das Ziel haben das Missverhältnis aus 0er und 1er abzumildern:

1. Oversampling: Datenzeilen mit der Minderheitenklasse in der Zielvariable werden so oft kopiert, dass das Verhältnis zwischen Mehrheits- und Minderheitenklasse ausgeglichener ist. Auswahl und Kopieren von Datenzeilen erfolgt häufig per Zufallsgenerator. Das Verfahren kommt eher bei kleineren Datensätzen zur Anwendung.
2. Undersampling: Es wird beim Trainieren nur ein Teil der Datenzeilen der Mehrheitsklasse verwendet, so dass das Verhältnis zwischen Mehrheits- und Minderheitenklasse ausgeglichener ist. Die Auswahl der verwendeten Datenzeilen erfolgt in der Regel per Zufallsgenerator. Das Verfahren kommt eher bei größeren Datensätzen zur Anwendung.

Hinweise zu Over- und Undersamplingverfahren:

1. Over- und Undersampling sollten nur auf dem Trainingsdatensatz durchgeführt werden, nicht aber auf dem Testdatensatz. (Speziell das Oversampling kann bei einer unsachgemäßen Durchführung zu schweren Fehlern führen, wenn Kopien von Datenzeilen sowohl im Trainings- als auch im Testdatensatz enthalten sind.)
2. Es gibt viele Varianten und Kombinationen von Under- und Oversamplingverfahren. Um weniger verschwenderisch mit den Daten umzugehen, kann beim Undersampling z.B. eine Technik ähnlich zur k -fachen Kreuzvalidierung angewendet werden: Dabei wird die Auswahl der Datenzeilen und das anschließende Trainieren des Algorithmus mehrfach durchgeführt, wobei jedesmal andere Datenzeilen verwendet werden. Bei jeder Durchführung j erhält man so eine andere Prognosefunktion $\hat{f}_j$. Die endgültige Prognosefunktion $\hat{f}$ ist dann der Durchschnitt der

Prognosefunktionen aus den verschiedenen Durchläufen

$$\hat{f} = \frac{1}{k} \sum_{j=1}^k \hat{f}_j.$$

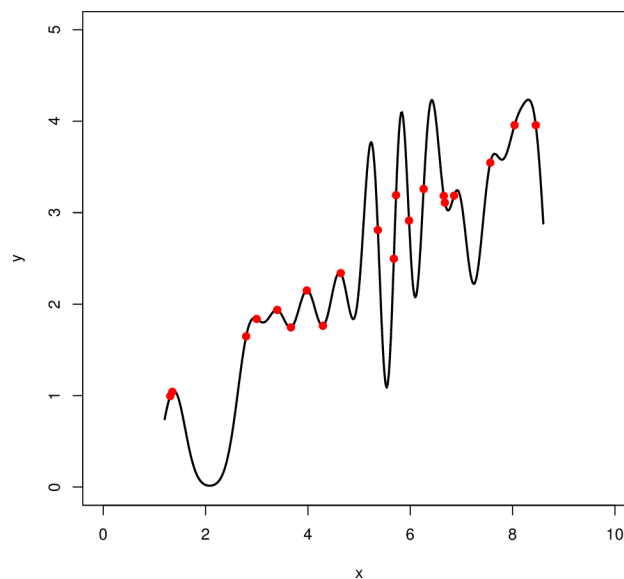
6 Tuning Maschinelles Lernverfahren

6.1 Einleitung: Warum müssen Maschinelle Lernverfahren getuned werden?

Maschinelle Lernverfahren zeichnen sich (in der Regel) dadurch aus, dass nur geringe Annahmen an den Modellzusammenhang

$$\text{Einflussvariablen} \xrightarrow{f} \text{Zielvariable}$$

gemacht werden. Dies kann aber zum Problem des Overfittings führen, wie folgendes Bild zeigt. Die Modellfunktion f passt “zu gut” zu den Daten, beschreibt aber nicht den allgemeinen Zusammenhang:

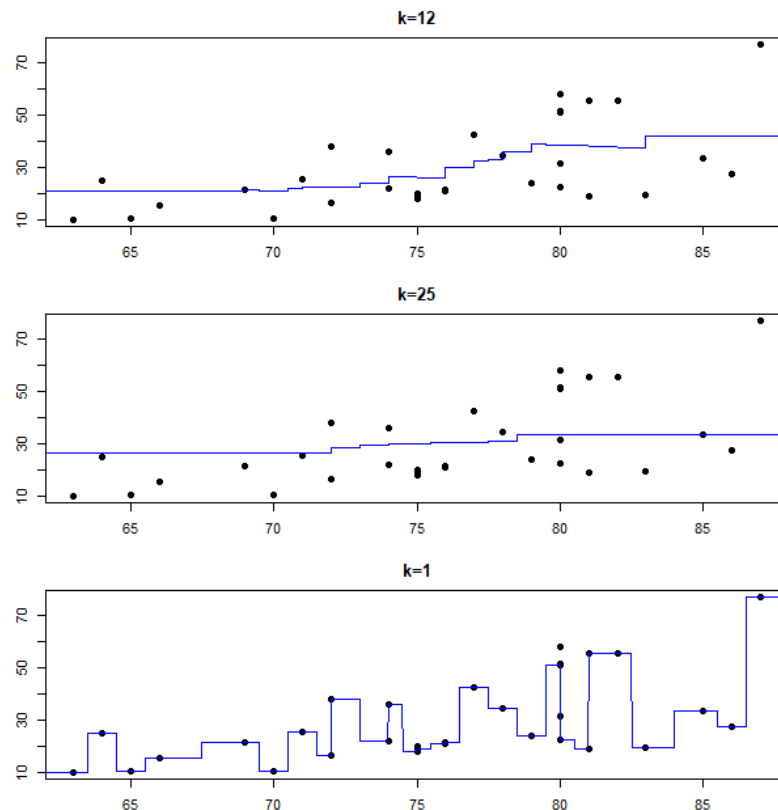


Man muss einen Kompromiss finden zwischen

- Wie gut soll die Funktion f die Trainingsdaten beschreiben?
- Wie kompliziert darf die Funktion f sein?

Je besser die Funktion die Trainingsdaten beschreibt, desto komplizierter wird die Funktion. Bei vielen Verfahren / Algorithmen gibt es einen oder mehrere Tuningparameter, der/die als Regler eingestellt werden können, um zu einem guten Kompromiss zu gelangen.

Beim k -Nearest-Neighbors-Verfahren ist dies die Anzahl k der nächsten Nachbarn. Folgende drei Bilder zeigen Ergebnisse für drei unterschiedliche Einstellungen:



Wie man an den Bildern sieht, erscheinen die Ergebnisse für $k = 12$ den Zusammenhang recht gut zu beschreiben. Für $k = 25$ erscheint die blaue Modellfunktion zu einfach zu sein, für $k = 1$ wiederum zu komplex. Leider gibt es keine Standardwerte oder einfache Faustregeln, wie Tuningparameter (wie hier das k) eingestellt werden sollte. Eine gute Einstellung der Tuningparameter hängt individuell vom Datensatz ab und muss daher auch datenbasiert bestimmt werden.

Im Supervised Learning können die Ergebnisse immer anhand der Prognosegüte validiert werden. Allerdings darf diese Validierung nicht einfach anhand der Trainingsdaten geschehen (siehe hierzu auch das Unterkapitel zu Trainings- und Testdaten). Würde man die Prognosegüte einfach anhand der

Trainingsdaten bestimmen, dann würde in den obigen Bildern der (extrem ungünstige) Wert $k = 1$ am besten abschneiden, da hier der Prognosefehler nahe null ist. Daher muss der Datensatz weiter aufgeteilt werden, um einen Validierungsdatensatz zur Wahl des Tuningparameters zu erhalten.

Folgendes Schema ist die einfachste Methode, die in der Praxis auch durchaus Anwendung findet. Dabei wird der ursprüngliche Datensatz dreigeteilt: Trainingsdaten, Validierungsdaten, Testdaten.

1. Trainingsdaten (z.B. 50% der Daten):

Auf diesem Datensatz wird – für verschiedene Werte der Tuningparameter – jeweils die Modellfunktion f berechnet.

2. Validierungsdaten (z.B. 20% der Daten):

Auf diesem Datensatz wird jeweils die Prognosegüte der Modellfunktionen f aus Schritt 1 berechnet. Der optimale Wert des Tuningparameters (bzw. bei mehreren Tuningparameter: die optimale Kombination von Werten der Tuningparameter) ist derjenige Wert (bzw. diejenigen Werte) mit optimaler Prognosegüte auf dem Validierungsdatensatz.

3. Testdaten (z.B. 30% der Daten):

Ist im Schritt 2 die optimale Wahl der Tuningparameter gefunden, dann wird die optimale Modellfunktion f nochmals mit diesen optimalen Tuningparametern berechnet. Hierzu werden die Trainings- und die Validierungsdaten wieder zu einem Datensatz zusammengefügt. Trainiert wird in diesem Schritt also auf diesem neu zusammengefügten Datensatz.

Anschließend wird die Prognosegüte auf dem Testdatensatz berechnet.

Das Vorgehen den Datensatz in drei Teile zu teilen, um einen eigenen Validierungsdatensatz zu erhalten, ist nicht besonders effizient und v.a. geeignet, wenn viele Daten zur Verfügung stehen. Um die vorhandenen Daten etwas besser auszunützen, wird dieses Vorgehen etwas modifiziert. Solche modifizierte Verfahren zur Wahl der Tuningparameter werden in den folgenden Unterkapiteln vorgestellt.

Zunächst listen wir aber für die bisher behandelten Maschinellen Lernverfahren die entsprechenden Tuningparameter auf:

- k -Nearest Neighbors:

Die Anzahl k der nächsten Nachbarn ist der einzige Tuningparameter.

- Entscheidungsbaum:

Bei Entscheidungsbäumen ist die Anzahl T an Endknoten eine gute Kennzahl für die Komplexität des Entscheidungsbaumes. Um einen guten Kompromiss zu finden, wird daher nach einem Entscheidungsbaum gesucht, der

$$\frac{(f_T(x_{1,1}, x_{1,2}, \dots, x_{1,k}) - y_1)^2 + \dots + (f_T(x_{n,1}, x_{n,2}, \dots, x_{n,k}) - y_n)^2}{n} + \lambda \cdot T$$

minimiert. Dabei ist der Parameter $\lambda > 0$ ein Tuningparameter. Dieser Parameter steuert, wie stark regularisiert werden soll, d.h. wie stark die Komplexität des Entscheidungsbaums bestraft werden soll.

- Neuronale Netze:

Tuningparameter sind die Anzahl der Hidden Layer (versteckte Schichten) und die Anzahl der Hidden Units (Neuronen) pro Hidden Layer. Je mehr Hidden Layer und je mehr Hidden Units eingesetzt werden, desto komplexer wird das Neuronale Netz.

6.2 Leave-One-Out

Leave-One-Out ist ein datenbasiertes Verfahren zur Wahl der Tuningparameter von Maschinellen Lernverfahren. Dabei wird die im vorherigen Unterkapitel beschriebene Dreiteilung des Datensatzes in Training, Validierung und Test modifiziert, um die Daten effizienter zu nutzen.

Das Vorgehen mit Leave-One-Out geht folgendermaßen:

1. Trainings- und Validierungsdaten (z.B. 70%):

Für verschiedene Werte t der Tuningparameter werden folgende Berechnungen durchgeführt: Von den Trainings- und Validierungsdaten wird ein einzelner Datenpunkt entfernt und auf den restlichen Datensatz wird die Modellfunktion f_t berechnet. Der um einen Datenpunkt verkleinerte Datensatz spielt also die Rolle der Trainingsdaten. Anschließend wird mit der so berechneten Modellfunktion f_t eine Prognose für den zuvor entfernten Datenpunkt berechnet und die Prognosegüte dieser Prognose berechnet. Der einzelne, zuvor entfernte Datenpunkt

spielt also die Rolle des Validierungsdatensatzes. Dies wird für jeden Datenpunkt des Trainings- und Validierungsdatensatzes wiederholt, so dass also jeder Datenpunkt einmal die Rolle des Validierungsdatensatzes spielt. Für jeden Datenpunkt erhält man also einen Wert für die Prognosegenauigkeit. Anschließend wird der Durchschnitt all dieser Prognosegenauigkeiten berechnet.

Der optimale Wert t für den Tuningparameter ist dann der Wert mit der besten durchschnittlichen Prognosegenauigkeit.

2. Testdaten (z.B. 30%):

Ist im Schritt 1 die optimale Wahl der Tuningparameter gefunden, dann wird die optimale Modellfunktion f nochmals mit diesen optimalen Tuningparametern berechnet. Hierzu wird der gesamte Trainings- und Validierungsdatensatz verwendet.

Anschließend wird die Prognosegüte auf dem Testdatensatz berechnet.

Das Leave-One-Out-Verfahren ist relativ einfach zu implementieren und führt zu sehr verlässlichen und stabilen Ergebnissen. Allerdings hat es den Nachteil, dass es sehr rechenintensiv ist: Für jeden Wert t des Tuningparameters muss das Maschinelle Lernverfahren sehr oft auf einem großen Datensatz berechnet werden. Werden beispielsweise nur 10 verschiedene Werte t des Tuningparameters durchprobiert und der Trainings- und Validierungsdatensatz aus 1000 Datenpunkten besteht, dann muss zur Validierung das maschinelle Lernverfahren

$10 \cdot 1000 = 10\,000$ mal auf einem Datensatz mit 999 Datenpunkten berechnet werden.

6.3 Kreuzvalidierung

Die k -fache Kreuzvalidierung ist eine Alternative zum Leave-One-Out, die nicht ganz so rechenintensiv ist. Die k -fache Kreuzvalidierung ist das Standardverfahren zur Wahl der Tuningparameter.

Das Verfahren funktioniert ähnlich wie Leave-One-Out, aber es wird zur Validierung nicht immer nur ein Datenpunkt, sondern eine Gruppe von Datenpunkten entfernt. Man könnte das Verfahren daher auch als “Leave-Several-Out” bezeichnen. Dadurch muss das maschinelle Lernverfahren viel weniger oft berechnet werden, so dass es deutlich weniger rechenintensiv ist.

Das Vorgehen der k -fachen Kreuzvalidierung wird im Folgenden für $k = 10$ vorgestellt. Die Wahl $k = 10$ ist der übliche Standardwert, aber auch $k = 5$ ist ein üblicher Wert.

1. Trainings- und Validierungsdaten (z.B. 70%):

Zunächst wird der Trainings- und Validierungsdatensatz zufällig in $k = 10$ Teile aufgeteilt. Für verschiedene Werte t der Tuningparameter werden dann folgende Berechnungen durchgeführt: Von den Trainings- und Validierungsdaten wird einer der $k = 10$ Teildatensätze entfernt und auf den restlichen 9 Teildatensätzen wird die Modellfunktion f_t berechnet. Der aus den restlichen 9 Teildatensätze zusammengesetzte Datensatz spielt also die Rolle der Trainingsdaten. Anschließend werden mit der so berechneten Modellfunktion f_t die Prognosen für den zuvor entfernten 10. Teildatensatz berechnet und die Prognosegüte dieser Prognosen berechnet. Der 10. Teildatensatz spielt also die Rolle des Validierungsdatensatzes. Dies wird für jeden der 10 Teildatensätze des Trainings- und Validierungsdatensatzes wiederholt, so dass also jeder Teildatensatz einmal die Rolle des Validierungsdatensatzes spielt. Für jeden der 10 Teildatensätze erhält man also einen Wert für die Prognosegenauigkeit. Anschließend wird der Durchschnitt dieser 10 Prognosegenauigkeiten berechnet.

Der optimale Wert t für den Tuningparameter ist dann der Wert mit der besten durchschnittlichen Prognosegenauigkeit.

2. Testdaten (z.B. 30%):

Ist im Schritt 1 die optimale Wahl der Tuningparameter gefunden, dann wird die optimale Modellfunktion f nochmals mit diesen optimalen Tuningparametern berechnet. Hierzu wird der gesamte Trainings- und Validierungsdatensatz verwendet.

Anschließend wird die Prognosegüte auf dem Testdatensatz berechnet.

Im Vergleich zu Leave-One-Out ist das Verfahren deutlich weniger rechenintensiv: Werden beispielsweise 10 verschiedene Werte t des Tuningparameters durchprobiert und der Trainings- und Validierungsdatensatz aus 1000 Datenpunkten besteht, dann muss zur Validierung das Maschinelle Lernverfahren

$$10 \cdot 10 = 100 \text{ mal auf einem Datensatz mit 900 Datenpunkten}$$

berechnet werden.

Bei den meisten (guten) Implementierungen maschineller Lernverfahren ist die Kreuzvalidierung bereits mit implementiert, so dass dies durch den Anwender nicht selbst gemacht werden muss. Bei der Anwendung müssen nur mögliche Werte t für die Tuningparameter vorgegeben werden, die bei der Kreuzvalidierung durchprobiert werden sollen.

7 Regularisierte Kern-basierte Verfahren (SVMs)

Die regularisierten Kern-basierten Verfahren bilden eine ganze Gruppe von sehr populären und leistungsfähigen maschinellen Lernverfahren. Das bekannteste dieser Verfahren sind die Support Vector Machines (SVM).

Das Modell

Wir betrachten wieder ein Regressionsmodell

$$y_i = f^*(x_{i,1}, x_{i,2}, \dots, x_{i,k}) + \varepsilon_i,$$

wobei y_i der Wert der Zielvariable ist und $x_{i,1}, x_{i,2}, \dots, x_{i,k}$ die Werte der Einflussvariablen sind; ε_i ist die Zufallsabweichung und i ist die Nummer des Datenpunktes (d.h. die Zeilenzahl in der Datentabelle).

Die Funktion f^* beschreibt den “wahren” Zusammenhang zwischen den Einflussvariablen und der Zielvariablen. Die Funktion f^* kann jede beliebige Form haben (linear, nichtlinear, U-förmig, etc.)

Ziel

Ziel der Datenanalyse ist es nun aus den Daten eine möglichst gute Schätzung für f^* zu berechnen.

Idee des Verfahrens

Es wird eine Kennzahl definiert, die für jede mögliche Funktion f angibt, wie gut sie den Zusammenhang zwischen Einfluss- und Zielvariablen beschreibt. Dann wird mittels mathematischer Optimierung die Funktion berechnet, die den besten Wert bei dieser Kennzahl hat.

Die Kennzahl besteht aus zwei Teilen:

- (i) dem sogenannten “Risiko”: eine Zahl, die beschreibt, wie gut die Funktion zu den Daten aus dem Datensatz passt
- (ii) der sogenannten “Regularisierung”: eine Maßzahl für die Komplexität der Funktion

Die Kennzahl hat dann die Form

$$\text{Kennzahl}(f) = \text{Risiko}(f) + \underbrace{\lambda \cdot \text{Komplexität}(f)}_{\text{Regularisierung}}$$

Hierbei ist λ ein Tuningparameter. Mit diesem Parameter kann gesteuert werden, wie stark regularisiert werden soll, d.h. wie stark die Komplexität der Funktion f bestraft werden soll. (“Einfache Funktionen werden bevorzugt!”) Die Regularisierung vermeidet das bereits besprochene Problem des Overfittings.

Details zum Verfahren

Das Risiko(f) ist eine Zahl, die beschreibt, wie gut die Funktion f zu den Daten aus dem Datensatz passt. Die Zahl wird zum Beispiel folgendermaßen berechnet:

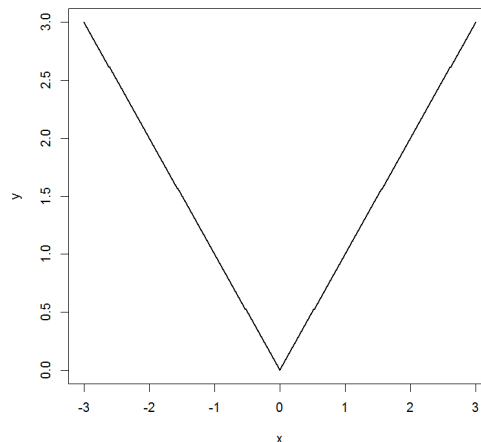
$$\begin{aligned}\text{Risiko}(f) &= \frac{|f(x_{1,1}, x_{1,2}, \dots, x_{1,k}) - y_1| + \dots + |f(x_{n,1}, x_{n,2}, \dots, x_{n,k}) - y_n|}{n} \\ &= \text{mittlere Abweichung der Prognosen von den tatsächlichen Daten}\end{aligned}$$

Alternativ kann für das Risiko auch die mittlere quadrierte Abweichung der Prognosen von den tatsächlichen Daten verwendet werden:

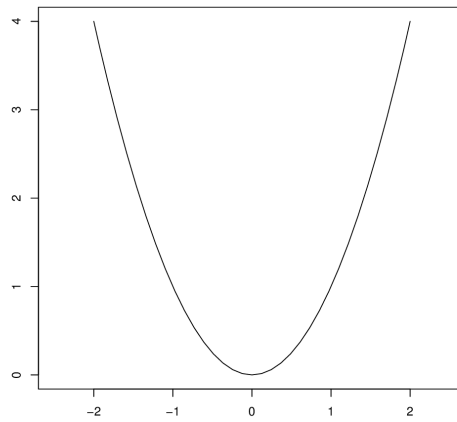
$$\text{Risiko}(f) = \frac{(f(x_{1,1}, x_{1,2}, \dots, x_{1,k}) - y_1)^2 + \dots + (f(x_{n,1}, x_{n,2}, \dots, x_{n,k}) - y_n)^2}{n}$$

Auf welche Weise das Risiko definiert wird, hängt von der verwendeten Verlustfunktion ab. Möglich sind zum Beispiel

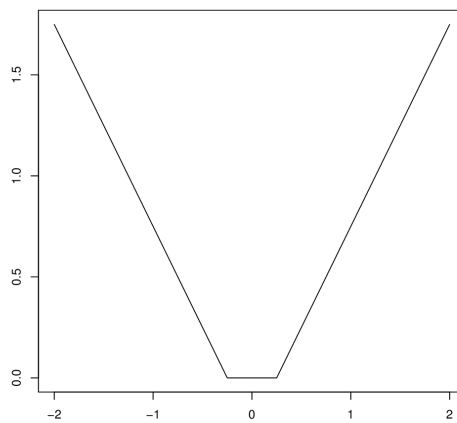
- die Betragsverlustfunktion



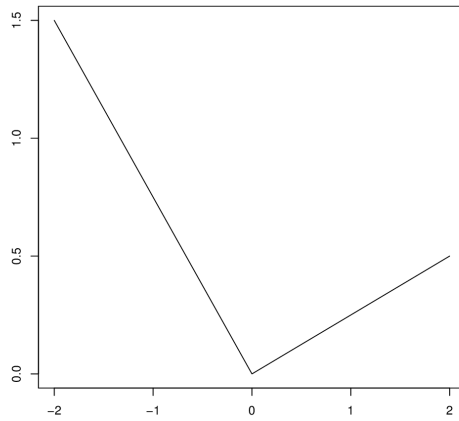
- die quadratische Verlustfunktion



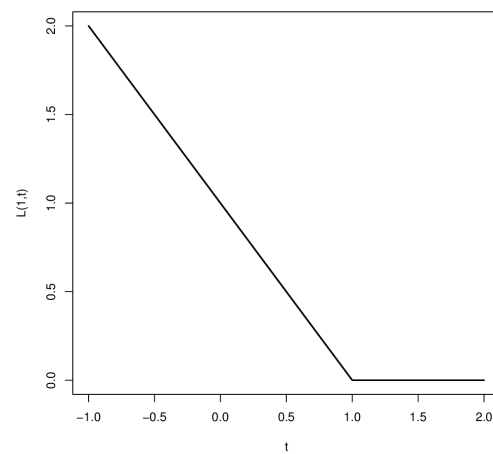
- die epsilon-insensitive Verlustfunktion



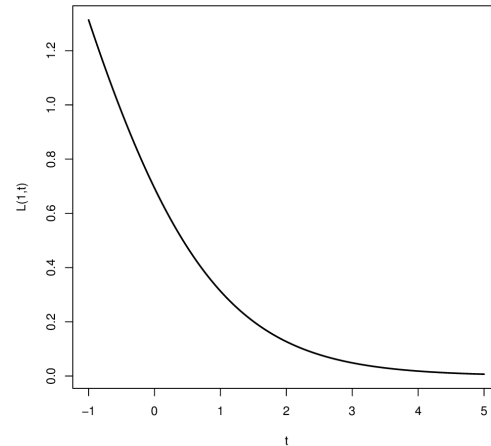
- die Pinball-Verlustfunktion



- die Hinge-Verlustfunktion



- die logistische Verlustfunktion



Die Wahl der Verlustfunktion hängt auch davon ab, welches überwachte Lernproblem bearbeitet wird:

- Regression
 - Betragsverlustfunktion
 - quadratische Verlustfunktion (Least-Squares Support Vector Regression)
 - epsilon-insensitive Verlustfunktion (Support Vector Regression)
- Quantils-Regression
 - Pinball-Verlustfunktion
- Klassifikation
 - Hinge-Verlustfunktion (Support Vector Machines)
 - logistische Verlustfunktion (Kernel Logistic Regression)

Tuning-Parameter

Jedes maschinelle Lernverfahren besitzt Tuningparameter. Bei den regularisierten Kern-basierten Verfahren sind das zwei Parameter

- der Regularisierungsparameter λ bzw. der Cost-Parameter C :

Dieser Parameter bestimmt, wie kompliziert die Funktion f sein darf. In manchen Software-Implementierungen muss man den Regularisierungsparameter λ angeben, in manchen Software-Implementierungen muss man den Cost-Parameter C angeben. (Der Cost-Parameter C entspricht $1/\lambda$.)

- der Kern-Parameter γ

Dieser Parameter bestimmt auch mit, welche Funktionen f in Frage kommen und hängt ebenfalls (indirekt) damit zusammen, wie kompliziert die Funktion f sein darf.

8 Universelle Konsistenz und Modellwahl

8.1 Universelle Konsistenz maschineller Lernverfahren

In diesem Unterkapitel müssen wir uns mit ein wenig Theorie befassen. Die theoretischen Feinheiten erfordern sehr viel abstrakte mathematische Konzepte wie Messbarkeit, Wahrscheinlichkeitsmaße, Integrationstheorie, Konvergenzarten. Für ein allgemeines Verständnis der für uns wesentlichen Ideen können und werden wir diese Feinheiten aber außer Acht lassen.

Modelle und Prognostizierbarkeit

Wir betrachten zunächst das Regressionsmodell

$$y_i = f^*(x_i) + \varepsilon_i,$$

wobei y_i der Wert der Zielvariable ist und x_i der Vektor mit den Werten der Einflussvariablen ist; ε_i ist die Zufallsabweichung und i ist die Nummer des Datenpunktes (d.h. die Zeilennummer in der Datentabelle).

Die Funktion f^* beschreibt den “wahren” Zusammenhang zwischen den Einflussvariablen und der Zielvariablen. Die Funktion f^* kann jede beliebige Form haben (linear, nichtlinear, U-förmig, etc.). In realen Problemen ist f^* unbekannt und es ist das Ziel des Supervised Learnings der wahren Funktion möglichst nahe zu kommen, da diese Funktion die bestmögliche Prognosefunktion für die Zielvariable y ist. Die Funktion f^* beschreibt aber nur die allgemeine Tendenz: Die Daten (x_i, y_i) folgen normalerweise nicht exakt diesem Zusammenhang f^* , sondern weichen (zufällig) davon ab. Der Wert ε_i stellt für jeden Datenpunkt (x_i, y_i) genau die Abweichung zwischen den realen Wert y_i der Zielvariablen und dem Wert $f^*(x_i)$ aus dem Modell dar:

$$\varepsilon_i = y_i - f^*(x_i).$$

Selbst wenn die wahre Funktion f^* bekannt wäre, wäre es daher nicht möglich die Werte der Zielvariablen y perfekt zu prognostizieren. Die Zielvariable y weicht normalerweise immer von $f^*(x)$ ab, da es die nicht-prognostizierbare Zufallsabweichung ε gibt.

Merke: Wenn die nicht-prognostizierbare Zufallsabweichung ε groß ist, dann ist es prinzipiell unmöglich ein System zu entwickeln, das genaue Prognosen liefert. In diesem Fall hilft es auch nicht weiter, wenn man Terrabyte-weise

zusätzliche Datenpunkte sammelt und viele Jahre an Entwicklungszeit investiert! Die Variabilität in den Daten ist zu groß, um genaue Prognosen erhalten zu können. Die einzige Möglichkeit ist, das Modell zu verbessern und das Ausmaß der nicht-prognostizierbaren Zufallsabweichungen zu verringern: das kann – wenn überhaupt – nur durch die Hinzunahme weiterer Einflussvariablen gelingen.

Beispiel: Aus den Einflussvariablen Alter, Geschlecht und Körpergröße lässt sich das Körpergewicht von Menschen nur sehr ungenau berechnen, weil Menschen sehr verschieden sind und durch Alter, Geschlecht und Körpergröße nicht genau genug beschrieben sind, um das Körpergewicht gut prognostizieren zu können. Besser werden die Prognosen jedoch, wenn man immer mehr Einflussvariablen berücksichtigt wie zum Beispiel Daten zum genauen Ernährungsverhalten, Daten zum Energieverbrauch, individuelle Genetik etc.

Wahres Risiko und Bayes-Risiko

Stehen in einem Supervised Learning Problem die betrachteten Einflussvariablen fest, dann würde die (unbekannte) Funktion f^* in dem Modell

$$y_i = f^*(x_i) + \varepsilon_i,$$

die bestmöglichen Prognosen für y basierend auf den in x zusammengefassten Einflussvariablen liefern.

Wie im Kapitel über Gütemaße gesehen, werden Prognosefunktionen f in einem Regresionsproblem durch Gütemaße der Form

$$\frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i)$$

bewertet, wobei L eine wählbare sogenannte Verlustfunktion ist. Beim MSE wird die quadratischen Verlustfunktion

$$L(f(x), y) = (f(x) - y)^2$$

gewählt, beim MAE die Betrags-Verlustfunktion

$$L(f(x), y) = |f(x) - y|.$$

In der praktischen Arbeit wird die Güte einer Prognosefunktion f basierend auf Testdaten $(x_{\text{test},i}, y_{\text{test},i})$ berechnet:

$$\mathcal{R}_{n_{\text{test}}}(f) = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} L(f(x_{\text{test},i}), y_{\text{test},i})$$

Dabei handelt es sich eigentlich nur um eine Schätzung der Prognosegüte, da der errechnete Wert $\mathcal{R}_{n_{\text{test}}}(f)$ ja von den (zufälligen) Testdaten abhängt. Man nennt den aus den Testdaten errechneten Wert $\mathcal{R}_{n_{\text{test}}}(f)$ das *empirische Risiko*. Je mehr Testdaten zur Verfügung stehen, desto genauer ist die Schätzung für die Prognosegüte: Das empirische Risiko nähert sich immer mehr dem wahren Risiko an:

$$\mathcal{R}_{n_{\text{test}}}(f) \xrightarrow{n_{\text{test}} \rightarrow \infty} \mathcal{R}_{\text{wahr}}(f).$$

Da die Funktion f^* die bestmöglichen Prognosen für y basierend auf den in x zusammengefassten Einflussvariablen liefert, hat f^* das kleinstmögliche wahre Risiko. Dieses kleinstmögliche wahre Risiko heißt *Bayes Risiko*. Somit gilt

$$\mathcal{R}_{\text{wahr}}(f^*) = \min_f \mathcal{R}_{\text{wahr}}(f) = \text{Bayes Risiko}.$$

Nun können wir uns der universellen Konsistenz zuwenden.

Universelle Konsistenz

Je mehr Trainingsdaten zur Verfügung stehen, desto besser sollten die Prognosen eines maschinellen Lernverfahrens sein. Ein maschinelles Lernverfahren f_n heißt *universell konsistent*, wenn es – bei wachsender Menge an Trainingsdaten – immer die bestmögliche Prognosegüte irgendwann (annähernd) erreicht:

$$\mathcal{R}_{\text{wahr}}(f_{n_{\text{train}}}) \xrightarrow{n_{\text{train}} \rightarrow \infty} \mathcal{R}_{\text{wahr}}(f^*) = \text{Bayes Risiko}$$

Eine besondere Betonung liegt dabei auf dem Wort “immer”. Das maschinelle Lernverfahren kann also die theoretisch bestmögliche Prognosegüte (Bayes Risiko) in jedem Fall erreichen⁷, egal wie komplex die Funktion f^* ist.

Bei der universellen Konsistenz handelt es sich um eine äußerst starke Eigenschaft, die die klassischen statistischen Verfahren (in der Regel) nicht besitzen. Lange Zeit war sogar unklar, ob es überhaupt ein maschinelles Lernverfahren geben kann, das universell konsistent ist. Das erste Verfahren, von dem mathematisch bewiesen wurde, dass es universell konsistent ist, ist das k -Nearest-Neighbor-Verfahren. Dies ist umso bemerkenswerter,

⁷Eigentlich müsste es “in fast jedem Fall” heißen. In der Regel müssen sehr geringe Modellannahmen getroffen werden, die aber nahezu keine praktische Bedeutung haben.

da es sich hierbei ja um ein sehr einfaches verfahren handelt. Heutzutage ist von vielen maschinellen Lernverfahren bekannt, dass sie universell konsistent sind. Richtig angewendet gehören dazu:

- k -Nearest Neighbor
- Entscheidungsbäume
- Neuronale Netze
- Regularisierte Kern-basierte Verfahren (insbesondere Support Vector Machines)

8.2 Modellwahl im maschinelles Lernen

8.2.1 Parametrische, semiparametrische und nichtparametrische Modelle

Die normale lineare Regression (Kleinste-Quadrate-Regression) ist nicht universell konsistent, da mit dem Verfahren nicht beliebig komplexe Modelle f gelernt werden können, sondern nur lineare Modelle. Innerhalb der linearen Modelle ist das Verfahren jedoch konsistent. Das heißt, mit wachsender Menge an Trainingsdaten nähert sich die Prognosegüte der linearen Regression immer mehr der bestmöglichen Prognose an, die durch ein lineares Verfahren erreicht werden kann:

$$\mathcal{R}_{\text{wahr}}(f_{n_{\text{train}}}) \xrightarrow{n_{\text{train}} \rightarrow \infty} \min_{f \in \mathcal{F}} \mathcal{R}_{\text{wahr}}(f)$$

Dabei ist z.B. bei nur einer Einflussvariablen

$$\mathcal{F} = \{f_{\beta_0, \beta_1} : x \mapsto \beta_0 + \beta_1 x \mid \beta_0, \beta_1 \in \mathbb{R}\}$$

die Menge aller linearen Funktionen. Es handelt sich dabei um ein parametrisches Modell, da die möglichen Funktionen f_{β_0, β_1} nur von (wenigen) Parametern β_j abhängen. Statt der Menge der linearen Funktionen kann das parametrische Modell z.B. aber auch aus der Menge der quadratischen Funktionen

$$\mathcal{F} = \{f_{\beta_0, \beta_1, \beta_2} : x \mapsto \beta_0 + \beta_1 x + \beta_2 x^2 \mid \beta_0, \beta_1, \beta_2 \in \mathbb{R}\}$$

bestehen. Im Gegensatz dazu ist bei einem nichtparametrischen Modell $\mathcal{F}$ die Menge aller möglichen (beliebig komplexen) Funktionen – oder zumindest eine sehr große (hochdimensionale) Menge an Funktionen. Das hat Vorteile, weil – sofern genügend Trainingsdaten zur Verfügung stehen – bei diesen nichtparametrischen Modellen bessere Prognosegenauigkeiten erreicht werden. Der Nachteil besteht aber darin, dass die Ergebnisse schlecht interpretierbar sind, weil sich bei mehreren Einflussvariablen komplizierte Funktionen

$$f : \mathbb{R}^m \rightarrow \mathbb{R}$$

nicht mehr grafisch darstellen lassen. (Hierbei ist m die Anzahl der Einflussvariablen.)

Einen guten Kompromiss zwischen parametrischen und nichtparametrischen Modellen bieten häufig semiparametrische Modelle, die “irgendwo dazwischen” liegen. Ein wichtiges Beispiel eines semiparametrischen Modells ist z.B. das additive Modell. Ein solches additives Modell bei m Einflussvariablen $x_1, \dots, x_m$ ist

$$\mathcal{F} = \{f : x \mapsto f_1(x_1) + f_2(x_2) + \dots + f_m(x_m) \mid f_1 \in \mathcal{F}_1, f_2 \in \mathcal{F}_2, \dots, f_m \in \mathcal{F}_m\}$$

Dabei ist $\mathcal{F}_j$ die Menge aller Funktionen $f_j : \mathbb{R} \rightarrow \mathbb{R}$ (oder zumindest eine sehr große hochdimensionale Funktionenmenge). Es sind auch Mischungen möglich, bei denen einzelne Komponenten $\mathcal{F}_j$ nichtparametrisch (große hochdimensionale Funktionenmengen) sind und andere Komponenten $\mathcal{F}_j$ parametrisch (z.B. alle linearen oder quadratischen Funktionen) sind. Der Vorteil von additiven Modellen ist, dass die einzelnen Komponentenfunktionen f_j grafisch darstellbar und daher interpretierbar sind, da es sich dabei ja nur um Funktionen $f : \mathbb{R} \rightarrow \mathbb{R}$ handelt.

Mit additiven Modellen sind zwar bessere Prognosegenauigkeiten erreichbar als mit parametrischen Modellen, aber nicht so gute wie mit nichtparametrischen Modellen (sofern genügend Trainingsdaten zur Verfügung stehen). Additive Funktionen der Form

$$f : (x_1, x_2, \dots, x_m) \mapsto f_1(x_1) + f_2(x_2) + \dots + f_m(x_m)$$

sind relativ speziell, wie folgendes Beispiel zeigt:

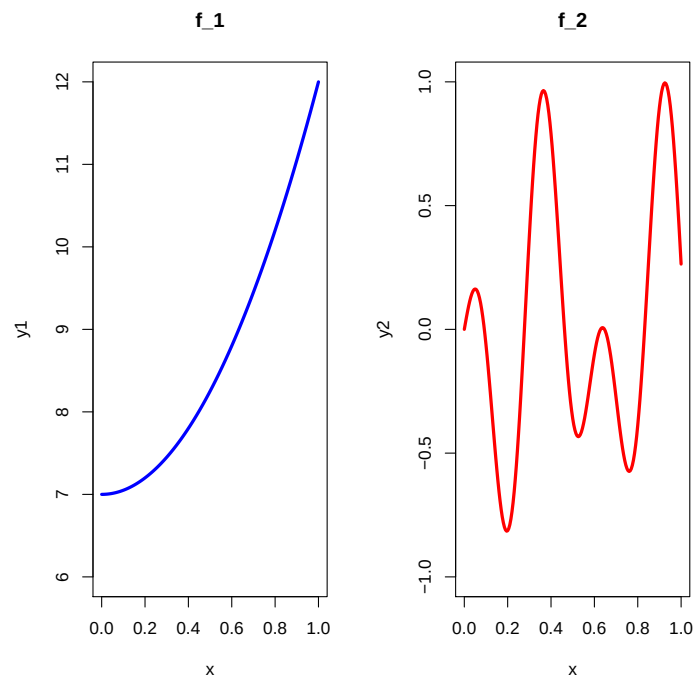
Die Funktion

$$f : \mathbb{R}^2 \rightarrow \mathbb{R}, \quad f(x_1, x_2) = \textcolor{blue}{f_1(x_1)} + \textcolor{red}{f_2(x_2)}$$

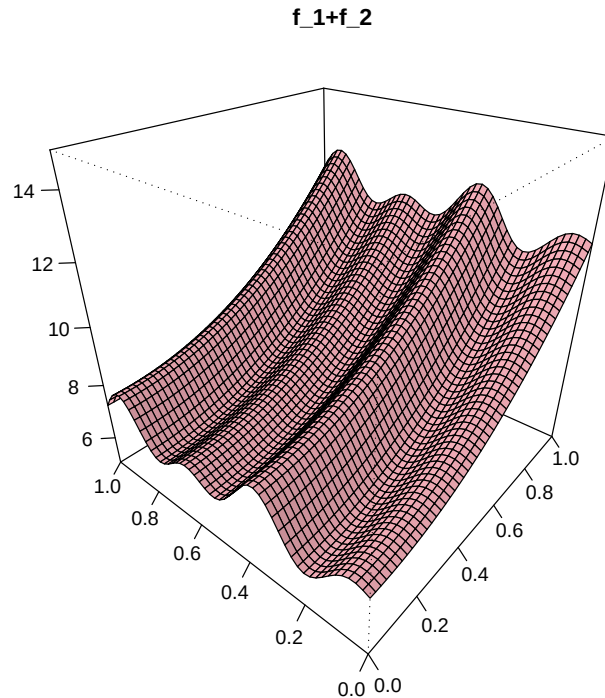
hat eine additive Struktur mit den beiden Komponentenfunktionen $f_1 : \mathbb{R} \rightarrow \mathbb{R}$ und $f_2 : \mathbb{R} \rightarrow \mathbb{R}$, wobei

$$f_1(x_1) = 7 + 5x_1^2 \quad \text{und} \quad f_2(x_2) = \sin(5x_2) \cos(17x_2).$$

Die beiden Komponentenfunktionen sind hier grafisch dargestellt:



Folgendes Bild zeigt die zusammengesetzte Funktion $f(x_1, x_2) = f_1(x_1) + f_2(x_2)$, man sieht die starke Regelmäßigkeit, der diese additiven Funktionen unterliegen:



Dies verdeutlicht die “Kosten” des Kompromisses. Durch die Verwendung additiver Modelle gewinnt man enorm an Interpretabilität, aber man schränkt sich auch deutlich bei der Komplexität ein – allerdings viel weniger als im Fall der linearen Regression. Ein weiterer Vorteil additiver Modelle ist auch, dass zum Trainieren additiver Modelle sehr viel weniger Daten benötigt werden als bei komplett nichtparametrischen Verfahren (wie Support Vector Machines oder Neuronalen Netzen). Es gibt verschiedene maschnelle Lernverfahren für additive Modelle⁸. Im nächsten Kapitel behandeln wir als Beispiel so ein maschinelles Lernverfahren, das sogenannte L_2 -Boosting.

⁸Ein solches maschinelles Lernverfahren basierend auf Support Vector Machines wurde z.B. entwickelt in:

Andreas Christmann & Robert Hable (2012): Consistency of support vector machines using additive kernels for additive models. *Computational Statistics & Data Analysis*, 56:854-873, 2012.

8.2.2 Beispiel: Additive Modelle mit L_2 -Boosting

Das Verfahren L_2 -Boosting ist nicht direkt über ein mathematisches Optimierungsproblem definiert, sondern über einen iterativen Algorithmus. Es ist für unsere Zwecke nicht nötig auf die (sehr umfangreichen) Details einzugehen, aber zumindest die Grundidee soll im Folgenden anhand der quadratischen Verlustfunktion

$$L(y, t) = (y - t)^2$$

kurz vorgestellt werden. Hierzu betrachten wir ein Regressionsproblem mit einer metrischen Zielvariablen und m Einflussvariablen. Dabei bezeichnet für jeden Datenpunkt (x_i, y_i) jeweils y_i den Wert der Zielvariablen im Datensatz und

$$x_i = (x_{i,1}, x_{i,2}, x_{i,3}, \dots, x_{i,m})$$

die Werte der Einflussvariablen. Das heißt, bei $x_{i,j}$ bezieht sich das i auf die Zeile im Datensatz und das j auf die Spalte im Datensatz.

Zur Berechnung einer Prognosefunktion $\hat{f}$ aus den Daten wird mit der konstanten Nullfunktion $\hat{f}^{(0)} = 0$ begonnen. In einem iterativen Verfahren wird diese Funktion dann schrittweise modifiziert. Dabei wird in jeder Iteration ℓ die Prognosefunktion aus dem letzten Schritt $\hat{f}^{(\ell-1)}$ zu einer neuen (komplexeren) Prognosefunktion $\hat{f}^{(\ell)}$ verfeinert. Dadurch erhält man eine Folge von Prognosefunktionen $\hat{f}^{(0)}, \hat{f}^{(1)}, \hat{f}^{(2)}, \hat{f}^{(3)}, \dots$, die immer komplexer werden. Damit die Prognosefunktion nicht zu komplex wird (Stichwort “Overfitting”), muss die Berechnung rechtzeitig gestoppt werden (“early stopping”). Im Folgenden ist aufgelistet, wie in jeder Iteration ℓ aus dem alten Stand der Prognosefunktion $\hat{f}^{(\ell-1)}$ die nächste, verfeinerte Version der Prognosefunktion $\hat{f}^{(\ell)}$ entsteht:

1. Es wird zunächst die (partielle) Ableitung der Verlustfunktion nach t berechnet:

$$L'(y, t) = \frac{\partial}{\partial t} L(y, t) = -2(y - t).$$

Aus dieser Ableitung, der alten Version der Prognosefunktion $\hat{f}^{(\ell-1)}$ und den (Trainings-)Daten werden dann Werte

$$u_i^{(\ell-1)} = -L'(y_i, \hat{f}^{(\ell-1)}(x_i)) = 2(y_i - \hat{f}^{(\ell-1)}(x_i)) \quad \text{für alle } i$$

berechnet.

2. Für jede Einflussvariable j werden nun separat die neuen Daten

$$(x_{1,j}, u_1^{(\ell-1)}), (x_{2,j}, u_2^{(\ell-1)}), (x_{3,j}, u_3^{(\ell-1)}), \dots (x_{n,j}, u_n^{(\ell-1)})$$

gebildet. Für jede Einflussvariable j entsteht dadurch ein neues, einfaches Regressionproblem mit nur einer Einflussvariablen $x_{i,j}$ und einer Zielvariablen $u_i^{(\ell-1)}$. Dieses Regressionsproblem wird für jedes j mit einem Regressionsalgorithmus gelöst. Der hierbei verwendete Regressionsalgorithmus kann im Grunde frei gewählt werden, aber es sollte sich dabei eher um ein einfaches Verfahren handeln. Wir nennen es daher einen *Weak Learner*, Detail dazu später. Mit Hilfe des Weak Learners wird also für jede Einflussvariable j eine Prognosefunktion $g_j^{(\ell-1)}$ berechnet:

$$\text{Einflussvariable } x_{i,j} \xrightarrow{g_j^{(\ell-1)}} \text{Zielvariable } u_i^{(\ell-1)}.$$

3. Da im vorherigen Schritt für jede Einflussvariable j ein Regressionsproblem gelöst wurde, haben wir nun also m Regressionsfunktionen

$$g_1^{(\ell-1)}, g_2^{(\ell-1)}, g_3^{(\ell-1)}, \dots, g_m^{(\ell-1)}$$

Nun wird anhand eines Gütekriteriums (z.B. MSE) für Regressionsprobleme ermittelt, welche dieser m Prognosefunktionen sein Regressionsproblem am besten löst.

Diese Regressionsfunktion $g_{j^*}^{(\ell-1)}$ und die dazugehörige Einflussvariable x_{i,j^*} wird nun ausgewählt und im nächsten Schritt weiterverwendet.

4. Nun wird die neue Prognosefunktion berechnet:

$$\hat{f}^{(\ell)}(x_{i,1}, x_{i,2}, \dots, x_{i,m}) = \hat{f}^{(\ell-1)}(x_{i,1}, x_{i,2}, \dots, x_{i,m}) + \nu g_{j^*}^{(\ell-1)}(x_{i,j^*}),$$

wobei $0 < \nu < 1$ eine vordefinierte Schrittweite ist.

Bemerkungen

- Abbruchkriterium:

Im Prinzip könnten bei diesem Algorithmus beliebig viele Iterationen ℓ durchgeführt werden. Wenn aber zu viele Iterationen durchgeführt werden, dann wird die Prognosefunktion $\hat{f}^{(\ell)}$ zu komplex und mit einer

zu hohen Anzahl an Iterationen entsteht das Problem des Overfittings. Der Algorithmus muss also rechtzeitig abgebrochen werden. (Dies bezeichnet man als “early stopping”.) Wenn jedoch andererseits zu wenig Iterationen durchgeführt werden, dann ist die Prognosefunktion $\hat{f}^{(\ell)}$ zu einfach und beschreibt die Daten nicht genau genug. Damit regelt die Anzahl der maximal durchgeführten Iterationen $\ell = \ell_{\text{stop}}$ die Komplexität der Prognosefunktion und dient somit als Tunig Parameter.

- Weak Learner

Im 2. Schritt jeder Iteration ℓ werden Prognosefunktionen $g_j^{(\ell-1)}$ mit einem Regressionsverfahren berechnet. Als Regressionsverfahren kommt hier nur ein einfaches Verfahren in Frage, dass das Regressionsproblem nicht besonders gut löst, ein sogenannter *Weak learner*. Die Idee dahinter ist: In jeder Iteration sollen jeweils nur kleine Fortschritte erzielt werden; große Fortschritte sollen nur durch die Durchführung vieler Iterationen erzielt werden.

Beim L_2 -Boosting werden als Weak Learner häufig Penalized B-Splines verwendet. Dabei handelt es sich um ein Verfahren, dessen Definition von der Struktur her ähnlich wie Least-Squares Support Vector Regression ist, wobei jedoch der Regularisierungsparameter λ so hoch eingestellt wird, dass die Komplexität sehr stark bestraft wird.

- Additive Struktur

Die Prognosefunktion $\hat{f}^{(\ell)}$ hat stets eine additive Struktur der Form

$$\hat{f}^{(\ell)}(x_{i,1}, x_{i,2}, \dots, x_{i,m}) = \hat{f}_1^{(\ell)}(x_{i,1}) + \hat{f}_2^{(\ell)}(x_{i,2}) + \dots + \hat{f}_m^{(\ell)}(x_{i,m}).$$

Diese additive Struktur entsteht dadurch, dass im 2. Schritt einer Iteration jedes $g_j^{(\ell-1)}$ nur von einer Einflussvariablen $x_{i,j}$ abhängt und im 3. Schritt die Funktion $g_{j^*}^{(\ell-1)}$ zu $\hat{f}^{(\ell-1)}$ hinzuaddiert wird.

- Variablenselektion

Das Verfahren führt immer auch automatisch eine Variablenselektion durch. Das heißt, das Verfahren nimmt in die Prognosefunktion nur die relevanten Einflussvariablen auf und entfernt die irrelevanten Einflussvariablen aus dem Modell. Dies geschieht dadurch, dass im 3. Schritt jeweils nur die beste Einflussvariable x_{i,j^*} ausgewählt wird. Wenn eine Einflussvariable $x_{i,j}$ keinen entscheidenden Beitrag für die Prognose

von y_i liefert, dann wird diese Variable niemals (in keiner Iteration ℓ) ausgewählt und taucht dadurch auch nicht in der resultierenden Prognosefunktion auf.

In der Praxis wird die Variablenselektion noch etwas verschärft: Auch wenn eine Variable nur sehr selten im 3. Schritt ausgewählt wird, dann wird sie nachträglich aus der Prognosefunktion entfernt.

- Tuning Parameter

Wie bereits im Punkt “Abbruchkriterium” erwähnt, stellt die Anzahl an maximal durchgeführten Iterationen ℓ_{stop} einen Tuning Parameter dar. Prinzipiell gäbe es aber noch viele andere potentielle Tuningparameter: die Schrittweite ν , der Regularisierungsparameter λ beim Penalized-B-Splines-Regressionsverfahren (Weak Learner) und noch weitere Parameter der Penalized B-Splines. Allerdings zeigt die Erfahrung, dass es in der Praxis ausreichend ist, alle diese potentiellen Tuning Parameter auf Standardwerte zu setzen und nur die maximale Anzahl der Iterationen ℓ_{stop} als Tuning Parameter zu wählen. Dieser kann dann wie üblich durch Kreuzvalidierung eingestellt werden.

9 Dimensionsreduktion

9.1 Einleitung: Der Fluch der hohen Dimension

In überwachten Lernproblemen ist es eine typische Situation, dass sehr viele potentielle Einflussvariablen vorliegen, aber nur wenige dieser potentiellen Einflussvariablen tatsächlich für die Zielvariable von Bedeutung sind. Es gibt maschinelle Lernverfahren, die aus den Daten die wesentlichen Einflussvariablen identifizieren können. Hierfür gibt es auch die Begriffe “Feature Selection” oder auf Deutsch “Variablenselektion”.

Bei z.B. 25 Einflussvariablen ist die Prognosefunktion eines überwachten Lernverfahrens eine Funktion

$$f : \mathbb{R}^{25} \rightarrow \mathbb{R}.$$

Wenn die wesentlichen Einflussvariablen identifiziert und die überflüssigen Einflussvariablen aus den Daten entfernt werden, dann reduziert sich die Anzahl der Dimensionen (Dimensionsreduktion). Wenn z.B. nur noch 8 wirklich wesentliche Einflussvariablen übrig bleiben, dann ist die Prognosefunktion im reduzierten überwachten Lernproblem nur noch eine Funktion

$$f : \mathbb{R}^8 \rightarrow \mathbb{R}.$$

Es gibt zwei Gründe, warum man die wesentlichen Variablen identifizieren möchte. Der erste Grund ist offensichtlich, der zweite ist nicht offensichtlich aber sehr schwerwiegend:

1. Werden die entscheidenden Einflussvariablen identifiziert, bedeutet dies einen erheblichen Erkenntnisgewinn. Man dann weiß dann, welche Variablen eine Rolle spielen und welche nicht. Dadurch weiß man, auf welche Faktoren künftig geachtet werden muss, welche Daten erhoben werden müssen und welche Daten nicht erhoben werden müssen.
2. Werden die überflüssigen Variablen entfernt, dann reduziert sich (wie oben gesehen) die Anzahl der Dimensionen im Lernproblem. Das Lernproblem wird einfacher.

Im Folgenden müssen wir uns den zweiten Grund etwas genauer ansehen. Überflüssige Variablen zu entfernen, um das Lernproblem zu vereinfachen, ist keine nette Verschönerungsmaßnahme, sondern essentiell. Der Grund dafür

ist, dass die Götter die Lernprobleme mit einem Fluch belegt haben, nämlich dem “Fluch der hohen Dimension”.

Um den Fluch der hohen Dimension zu verstehen, betrachten wir folgendes Beispiel mit $m = 25$ verschiedenen Einflussvariablen und $n = 1\,000\,000$ Datenpunkten. Nehmen wir an, dass die Einflussvariablen unabhängig voneinander sind und zufällig verteilte Werte zwischen 0 und 1 haben (Gleichverteilung). Die 1 000 000 Datenpunkte sind also einigermaßen gleichmäßig über einen 25-dimensionalen Würfel mit Kantenlänge 1 und Volumen 1 verteilt.

Betrachten wir in dem Würfel nun einen Teilwürfel mit Kantenlänge 0,5. Wie hoch ist die Wahrscheinlichkeit, dass sich in dem Teilwürfel mit Kantenlänge 0,5 zufällig kein einziger der 1 000 000 Datenpunkte befindet?

Wenn die Dimension wächst, dann vereinzeln die Datenpunkte zunehmen – und das schon bei relativ niedrigen Dimensionszahlen und sehr vielen Datenpunkten. Jedes maschinelle Lernverfahren zur Lösung eines überwachten Lernproblems lebt davon, dass der Raum dicht mit Trainingsdaten aufgefüllt ist. Wenn die Dimension (Anzahl an Einflussvariablen) größer wird, dann ist das aber nicht mehr möglich. Der Raum wird immer leerer und die Datenpunkte sind so einsam wie ein Professor in der Sprechstunde außerhalb der Prüfungszeiten.

Sehen wir uns hierzu noch ein paar andere Zahlen an. Wir verteilen n Datenpunkten völlig gleichmäßig als Gitterpunkte über einen m -dimensionalen Würfel. Wie groß muss die Anzahl n dann sein, damit der Raum so “gut” abgedeckt ist, dass für jeden beliebigen Punkt, der Abstand zum nächsten Gitterpunkt höchstens 0,25 beträgt:

	$m = 1$	$m = 5$	$m = 10$	$m = 25$
n	2	32	1024	33 554 432

Der Abstand 0,25 klingt vielleicht nicht so groß, aber er ist riesig, wenn man bedenkt, dass der Würfel nur Kantenlänge 1 hat. Der Abstand beträgt ein Viertel der gesamten Spannweite des Würfels!

Wenn der Raum nicht dicht genug mit Datenpunkten gefüllt ist, dann kann das maschinelle Lernverfahren die Zusammenhänge nicht gut lernen. Es ist

daher nötig die Dimension so klein wie möglich zu halten, indem nur so viele Einflussvariablen, wie wirklich nötig, verwendet werden.

Um die Anzahl der Variablen zu reduzieren gibt es grundsätzlich zwei Möglichkeiten:

1. Neubildung von Variablen: Aus den ursprünglichen Variablen werden neue Variablen konstruiert, so dass die ursprünglich in vielen Variablen enthaltene Information auf wenige Variablen verdichtet werden. Der Hauptvertreter dieser Art von Verfahren ist die Principle Component Analysis (PCA) oder auf Deutsch “Hauptkomponentenanalyse”. Dieses Verfahren kann auch außerhalb eines überwachten Lernproblems verwendet werden und ist ein klassisches Verfahren der Statistik.
2. Entfernen von Variablen (Variablenselektion): In einem unüberwachten Lernproblem werden diejenigen Variablen entfernt, die keinen oder nur einen geringen Zusammenhang mit der Zielvariable haben. Hierzu gibt es eine Reihe von Verfahren. Beispiele für solche Verfahren sind:
 - L_2 -Boosting (Kapitel 8.2.2)
 - Entscheidungsbäume: Da beim Aufbau des Baumes schrittweise die wichtigsten Variablen ausgewählt werden, führt das Verfahren implizit eine Variablenselektion durch. Unwichtige Variablen tauchen im Baum nicht auf (sofern durch geeignetes Tuning das Overfitting vermieden wird und die Anzahl der Endknoten nicht zu hoch ist).
 - LASSO (Kapitel 9.2)

9.2 LASSO

Verfahren:

Wie die gewöhnliche lineare Regression baut auch LASSO auf einem linearen Modell auf:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \dots + \beta_m x_{i,m} + \varepsilon_i.$$

Dabei bezieht sich in $x_{i,j}$ wieder der Index i auf die Zeilennummer und j auf die Spalte (Einflussvariablen) im Datensatz. Somit ist also m die Anzahl der Einflussvariablen.

Zur Berechnung der Regressionskoeffizienten $\beta_0, \beta_1, \dots, \beta_m$ wird bei der gewöhnlichen linearen Regression die Summe (oder – was auf das Gleiche hinausläuft – der Durchschnitt) der quadrierten Abstände zwischen den wahren Werten der Zielvariable y_i und den Werten aus dem Modell $\beta_0 + \beta_1 x_{i,1} + \dots + \beta_m x_{i,m}$ minimiert:

$$\min_{\beta_0, \beta_1, \dots, \beta_m} \frac{1}{n} \sum_{i=1}^n \left(y_i - (\beta_0 + \beta_1 x_{i,1} + \dots + \beta_m x_{i,m}) \right)^2$$

LASSO ist sehr ähnlich aufgebaut. Der einzige Unterschied ist, dass nun noch ein Strafterm hinzukommt, der die Größe der Parameter β_j bestraft:

$$\min_{\beta_0, \beta_1, \dots, \beta_m} \frac{1}{n} \sum_{i=1}^n \left(y_i - (\beta_0 + \beta_1 x_{i,1} + \dots + \beta_m x_{i,m}) \right)^2 + \lambda \cdot \sum_{j=0}^m |\beta_j|$$

Dieser Strafterm $\sum_{j=0}^m |\beta_j|$ führt dazu, dass die Parameter β_j tendenziell kleiner sind (Schrumpfen / Shrinkage) als bei der gewöhnlichen linearen Regression und einige Parameter β_j sogar auf Null gesetzt werden (Auswahl / Selection). Daher auch der Name des Verfahrens:

Least Absolute Shrinkage and Selection Operator (LASSO)

Man beachte dabei: Wenn ein Parameter β_j auf Null gesetzt wird, dann ist das gleichbedeutend damit, dass die entsprechende Einflussvariable $x_{i,j}$ aus dem Modell entfernt wird.

Der Aufbau des Verfahrens ist ähnlich zu Support Vector Machines (SVM) und Entscheidungsbäumen. Bei LASSO bestraft der Strafterm $\sum_{j=0}^m |\beta_j|$ auch die Komplexität des Modells, denn je größer die Parameter β_j sind und je mehr Parameter β_j ungleich Null sind (also je mehr Einflussvariablen im Modell behalten werden), desto komplexer ist das Modell. Wie bei Support Vector Machines und Entscheidungsbäumen ist $\lambda > 0$ ein Tuning Parameter der die Modellkomplexität regelt. Je größer der Wert von λ eingestellt wird, desto mehr Parameter β_j werden auf Null gesetzt und desto mehr Einflussvariablen werden aus dem Modell entfernt.

Verwendung in der Praxis:

LASSO kann als Alternative zu anderen überwachten Lernverfahren verwendet werden wie zum Beispiel der gewöhnlichen linearen Regression, Entscheidungsbäumen, Support Vector Regression, Neuronalen Netzen ... In der Praxis wird LASSO meistens aber nur in einem vorgeschalteten Schritt zur Variablenselektion verwendet. Die endgültige Prognosefunktion wird dann häufig mit einem anderen Verfahren (basierend auf den von LASSO ausgewählten Einflussvariablen) berechnet. Es ergibt sich also ein zweistufiges Vorgehen:

1. Zunächst wird LASSO unter Verwendung aller potentieller Einflussvariablen berechnet. Diejenigen Einflussvariablen, die von LASSO nicht ausgewählt wurden (also deren Regressionskoeffizient β_j von LASSO auf Null gesetzt wurde), werden anschließend aus dem Datensatz entfernt.
2. Im zweiten Schritt wird ein beliebiges anderes überwachtetes Lernverfahren verwendet, um eine Prognosefunktion basierend auf den verbliebenen Einflussvariablen zu berechnen.

Natürlich müssen weiterhin in jedem der beiden Schritte die entsprechenden Tuningparameter durch Kreuzvalidierung (oder ein anderes Tuning-Verfahren) bestimmt werden.

Bei der Kreuzvalidierung kann bei LASSO ein unschöner Effekt eintreten: Da bei der Kreuzvalidierung der Trainingsdatensatz zufällig in Blöcke aufgeteilt wird, spielt der Zufall eine Rolle. Es kann vorkommen, dass man für ein und den selben Datensatz jedesmal leicht andere Ergebnisse erhält, wenn man die Variablenselektion mit LASSO durchführt. Es werden also nicht immer diesselben Variablen ausgewählt. Um diesen Zufallseffekt zu vermeiden, ist es nicht unüblich, die Variablenselektion sehr oft durchzuführen (z.B. 100 mal) und sich dann anzusehen, welche Variable in den 100 Durchläufen wie oft ausgewählt wurde. Basierend auf diesen Ergebnissen wird dann entschieden, welche Variablen im zweiten Schritt für das überwachte Lernproblem weiter verwendet werden.

Achtung: LASSO beruht auf einem linearen Regressionsmodell. Die Variablenselektion funktioniert daher gut bei monotonen Zusammenhängen zwischen Einflussvariablen und Zielvariable, aber das Verfahren funktioniert nicht gut bei nicht-monotonen (z.B. U-förmigen) Zusammenhängen. Bei nicht-monotonen Zusammenhängen kann zum Beispiel das L_2 -Boosting verwendet werden, da das L_2 -Boosting nicht auf einem linearen Modell, sondern auf einem additiven Modell beruht.

10 Zeitreihen und Online-Learning

10.1 Einleitung

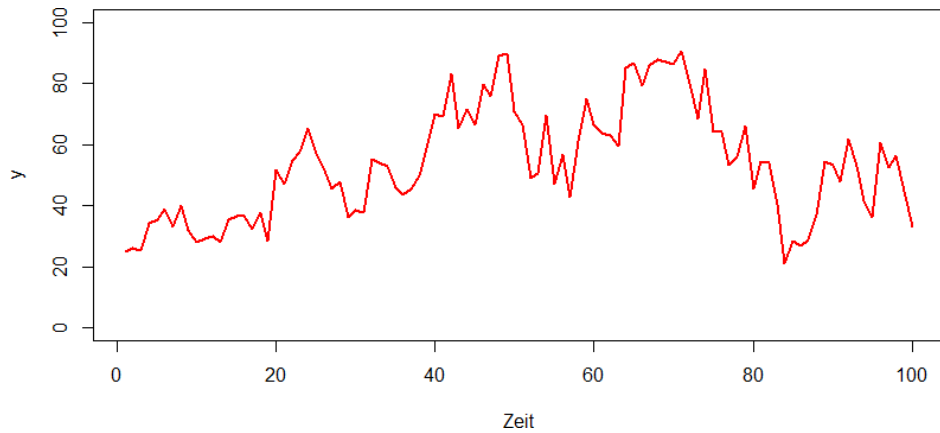
In den bisherigen Kapiteln sind wir stets davon ausgegangen, dass der Wert y_i der Zielvariable eines Datenpunktes nur von den Werten der Einflussvariablen $x_{i,1}, x_{i,2}, \dots, x_{i,m}$ abhängt – nicht aber von den anderen Werten $y_1, y_2, \dots, y_{i-1}$ der Zielvariable. Dies ist zum Beispiel der Fall, wenn es sich um Patientendaten handelt, bei denen jeder Datenpunkt einem Patienten entspricht. Die Zustände der Patienten beeinflussen sich (in der Regel) nicht gegenseitig. Ein anderes Beispiel sind Produktionsdaten: Ein Datenpunkt entspricht einem Produktionsauftrag, bei dem die Zielvariable y_i einen Wert für die Qualität des produzierten Teils angibt und die Einflussvariablen $x_{i,j}$ die jeweiligen Maschineneinstellungen enthalten. Auch hier ist es plausibel, dass sich die Werte von $y_1, y_2, \dots, y_n$ kaum gegenseitig beeinflussen. Es gibt aber viele Fälle, in denen die Werte der Zielvariable sehr wohl zusammenhängen.

Betrachten wir als Beispiel tägliche Aktienkurse eines Unternehmens: Die Zielvariable y_i sei der Schlusskurs einer bestimmten Aktie am Tag i . Der Aktienkurs y_i am Tag i hängt sicherlich sehr stark vom Aktienkurs y_{i-1} des Vortages $i-1$ ab! Das sieht man auch daran, dass vielfach der Aktienkurs y_{i-1} des Vortages als beste Prognose für den Aktienkurs y_i von heute angesehen wird.

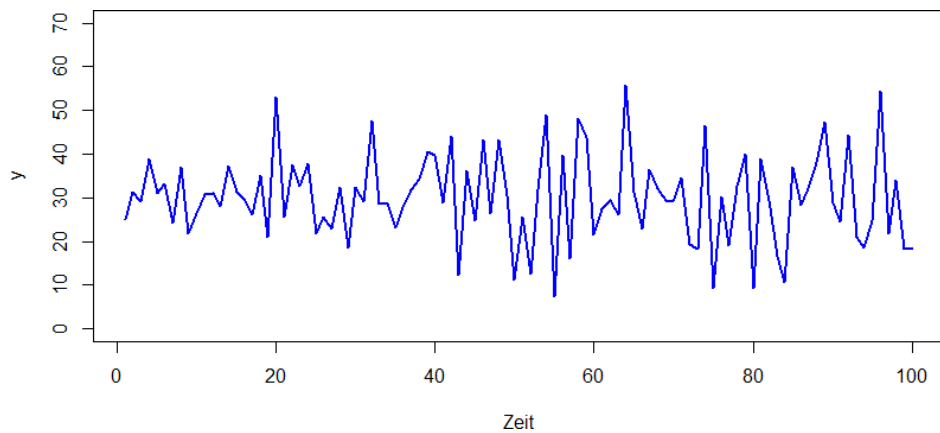
Wenn es sich bei den Datenpunkten um den zeitlichen Verlauf eines Wertes handelt, dann nennt man das eine *Zeitreihe*. In den folgenden beiden Abbildungen sehen Sie typische Verläufe von Zeitreihen, wenn die vorherigen Werte **einen Einfluss** auf nachfolgende Werte haben (**oberes** Bild), und wenn die vorherigen Werte **keinen Einfluss** auf nachfolgende Werte haben (**unteres** Bild).

Besispiele für Zeitreihen, bei denen vorherige Werte Einfluss auf nachfolgende Werte haben (**rote Beispielgrafik**):

- Kursverläufe von Aktien, Währungen, ...
- Temperaturverläufe
- Verläufe von Sensorwerten in einer Maschine



Rote Linie: Vorherige Werte haben Einfluss auf nachfolgende Werte wie zum Beispiel bei Kursverläufen von Aktien.



Blaue Linie: Vorherige Werte haben keinen Einfluss auf nachfolgende Werte wie zum Beispiel bei der Höhe der Einkaufssummen von aufeinander folgenden Kunden im Supermarkt.

- Verläufe der Gesamt-Tagesverkäufe in einem Geschäft

Abhängigkeiten zwischen vorherigen und nachfolgenden Werten lassen sich zum Beispiel durch die Korrelation messen. Wenn die Werte y_i in einer Zeitreihe mit vorangegangenen Werten (z.B. y_{i-1}) korrelieren, dann wird dies als Autokorrelation bezeichnet. Die Korrelation zwischen den Werten y_i mit den jeweils vorherigen Werten y_{i-1} beträgt in den Beispielgrafiken

- **Rote Kurve:** $\text{cor}(y_i, y_{i-1}) = 0,84$
- **Blaue Kurve:** $\text{cor}(y_i, y_{i-1}) = -0,28$

Wenn in einer Zeitreihe ein stärkeren Zusammenhang zwischen y_i mit vorangegangenen Werten besteht, dann sollte dies berücksichtigt werden, um gute Prognosen zu erzielen. Hierfür gibt es verschiedene Techniken, von denen wir einige in den nachfolgenden Unterkapiteln behandeln werden.

Eine typische Anwendung bei Zeitreihen ist es basierend auf den vergangenen Werten $y_1, y_2, \dots, y_i$ und den Werten für die Einflussvariablen den Wert der Zielvariable y_i für den nächsten Zeitpunkt vorherzusagen. Weil dabei kontinuierlich im Zeitverlauf neue Datenpunkte eintreffen und der jeweils nächste Wert für die Zielvariable vorhergesagt werden soll, wird dies im maschinellen Lernen auch als *Online Learning* bezeichnet.

Nachfolgende Tabelle soll noch einmal die typische Aufgabenstellung beim Online Learning verdeutlichen. Dabei sind die für die Prognose der Zielvariable y_i im Zeitpunkt i bekannten Werte in blau markiert. (Die blauen Werte können also für die Prognose verwendet werden.) Der unbekannte und zu prognostizierende Wert y_i im Zeitpunkt i ist in rot markiert. Die sonstigen zum Zeitpunkt der Prognose ebenfalls unbekannten Werte sind in schwarz aufgeführt. Die schwarzen Werte können also nicht für die Prognose verwendet werden. Bitte behalten Sie dieses Schema gut im Kopf und beachten Sie immer, welche Werte zum Zeitpunkt der Prognose bekannt sind (und daher verwendet werden können) und welche nicht bekannt sind (und daher nicht verwendet werden können)!

Zeitpkt.	Einflußvariablen				Zielvar.
1	$x_{1,1}$	$x_{1,2}$	$\dots$	$x_{1,m}$	y_1
2	$x_{2,1}$	$x_{2,2}$	$\dots$	$x_{2,m}$	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$i - 1$	$x_{i-1,1}$	$x_{i-1,2}$	$\dots$	$x_{i-1,m}$	y_{i-1}
i	$x_{i,1}$	$x_{i,2}$	$\dots$	$x_{i,m}$	y_i
$i + 1$	$x_{i+1,1}$	$x_{i+1,2}$	$\dots$	$x_{i+1,m}$	y_{i+1}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
n	$x_{n,1}$	$x_{n,2}$	$\dots$	$x_{n,m}$	y_n

10.2 Beispiel: Nachfragedaten

Als Beispiel verwenden wir in den nachfolgenden Unterkapiteln wöchentliche Nachfragedaten für ein bestimmtes Produkt für einen Zeitraum von 4 Jahren (also $4 \cdot 52 = 208$ Datenpunkte).

Der Datensatz besitzt in seiner Rohform folgende Spalten:

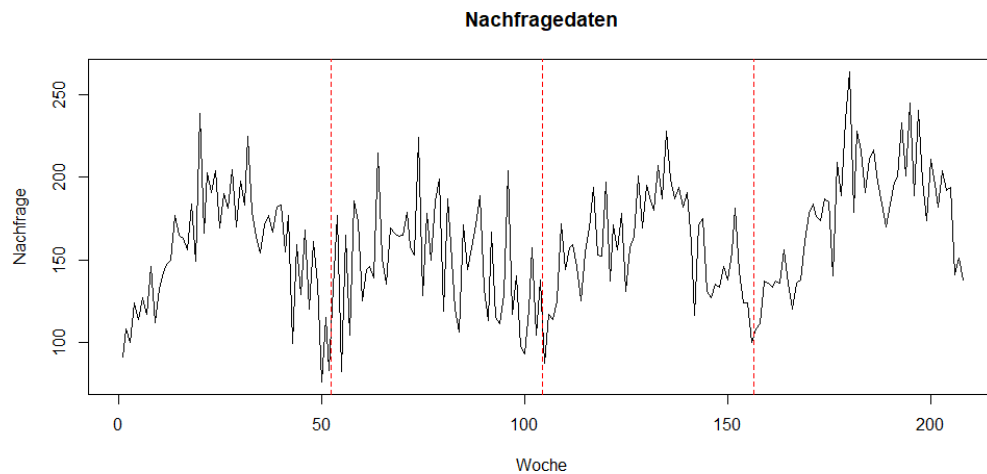
- Jahr: eine Index-Zahl für das Jahr (1, 2, 3, 4)
- KW: die Nummer der Kalenderwoche (1, 2, 3, $\dots$, 52)
- Preis: der Preis in Euro
- Nachfrage: die Gesamtnachfrage der jeweiligen Woche (in Stück)

Nachfolgende Tabelle zeigt die ersten Zeilen des Datensatzes in seiner Roh-

form:

Nachfrage-Rohdaten			
Jahr	KW	Preis	Nachfrage
1	1	2.00	91
1	2	2.09	108
1	3	2.40	100
1	4	2.57	124
1	5	2.24	114
⋮	⋮	⋮	⋮
2	1	1.40	133
2	2	1.50	177
2	3	1.20	82
⋮	⋮	⋮	⋮
4	52	1.42	138

Der zeitliche Verlauf der Nachfrage ist in folgendem Bild aufgezeichnet, die 4 Jahre sind durch die roten Striche getrennt.



Nachfragedaten: wöchentlicher Verlauf der Nachfrage für ein bestimmtes Produkt für einen Zeitraum von 4 Jahren (also $4 \cdot 52 = 208$ Datenpunkte). Die 4 Jahre sind durch die roten Striche getrennt

Ein einfaches Regressionsmodell für die Nachfrage würde folgendermaßen aussehen:

- Einflussvariable: Preis
- Zielvariable: Nachfrage

Wie man am zeitlichen Verlauf sieht, ist die Nachfrage jedoch autokorreliert: die Nachfrage in der Woche i hängt deutlich von der Nachfrage in der Vorwoche $i - 1$ ab. Der Korrelationskoeffizient zwischen der Nachfrage und der Nachfrage der Vorwoche beträgt

$$\text{cor}(y_i, y_{i-1}) = 0,51.$$

An der Grafik sieht man bei genauerer Betrachtung auch, dass die Nachfrage saisonalen Schwankungen unterliegt: im Sommer scheint die Nachfrage höher zu sein als im Winter. Darüber hinaus scheint es über die Jahre einen leicht steigenden Trend in der Nachfrage zu geben. Das eben erwähnte einfache Regressionsmodell spiegelt all dies nicht wieder und wird daher nur zu schlechten Regressions- und Prognoseergebnissen führen. Das Modell muss daher erweitert werden, um die zeitlichen Effekte und Abhängigkeiten geeignet zu berücksichtigen.

Die Berücksichtigung von Trend- und Saisoneffekten ist die gängigste Methode bei der Analyse von Zeitreihen und wird im folgenden Unterkapitel vorgestellt.

10.3 Trend und Saisonkomponenten

Lineare Effekte

Im Beispiel aus dem Unterkapitel 10.2 wäre ein einfaches lineares Regressionsmodell

$$\text{Nachfrage} = \beta_0 + \beta_1 \cdot \text{Preis} + \text{Zufallsfehler}$$

Wie im Unterkapitel 10.2 erwähnt, würde dies aber zu schlechten Ergebnissen führen, da dieses Modell die vorliegenden zeitlichen Effekte und Abhängigkeiten nicht berücksichtigt. Ein möglicherweise vorhandener linearer Trend kann durch die Hinzunahme der Zeit als weitere Einflussvariable integriert werden:

$$\text{Nachfrage} = \beta_0 + \beta_1 \cdot \text{Preis} + \beta_2 \cdot \text{Zeit} + \text{Zufallsfehler}$$

Zusätzlich können saisonale Effekte durch die Bildung einer weiteren Variable integriert werden. Eine solche Variable, könnte zum Beispiel die Variable “Saison” sein, in der vermerkt wird, ob die jeweilige Kalenderwoche im Frühling, Sommer, Herbst oder Winter liegt. Hierbei handelt es sich um eine kategorielle Variable mit 4 verschiedenen Kategorien. Bei der linearen Regression wird eine solche kategorielle Variable in 3 neue sogenannte Dummy-Variablen (0-1-Variablen) zerlegt, z.B. in:

- Sommer: nein/ja (0/1)
- Herbst: nein/ja (0/1)
- Winter: nein/ja (0/1)

Zwei Dinge sind hierbei zu beachten. Erstens: Es gibt keine eigene Dummy-Variable für Frühling, denn wenn die Kalenderwoche weder im Sommer, noch im Herbst, noch im Winter liegt, dann geht hieraus bereits hervor, dass die Kalenderwoche im Frühling liegen muss. Frühling wäre in diesem Beispiel also die Basiskategorie. Zweitens: In den meisten Software-Implementierungen (so auch in R) geschieht die Bildung der Dummy-Variablen intern unter der Oberfläche und muss nicht vom Nutzer selbst durchgeführt werden - sofern die Variable korrekt als kategorielle Variable gekennzeichnet ist.

Das vollständige Regressionsmodell mit allen Regressionskoeffizienten β_j lautet dann:

$$\begin{aligned} \text{Nachfrage} = & \beta_0 + \beta_1 \cdot \text{Preis} + \beta_2 \cdot \text{Zeit} + \\ & + \beta_3 \cdot \text{Sommer} + \beta_4 \cdot \text{Herbst} + \beta_5 \cdot \text{Winter} + \\ & + \text{Zufallsfehler} \end{aligned}$$

Die Interpretation der Saison-Koeffizienten β_3 , β_4 und β_5 ist dabei wie üblich bei den Regressionskoeffizienten für kategorielle Variablen. Der Koeffizient β_3 für Sommer bedeutet hier beispiesweise: unter ansonsten gleichen Bedingungen ist die Nachfrage im Sommer im Mittel um den Wert β_3 höher als im Frühling (Basiskategorie). Dabei kann der Wert β_3 natürlich auch negativ sein, was bedeuten würde, dass dann die Nachfrage im Sommer niedriger als im Frühling wäre.

Um einen linearen Trend und Saisonkomponenten wie in obigem Modell zu integrieren, muss der Datensatz entsprechend vorbereitet werden. Der aus

den Rohdaten aufbereitete Beispieldatensatz aus Unterkapitel 10.2 würde dann zum Beispiel so aussehen:

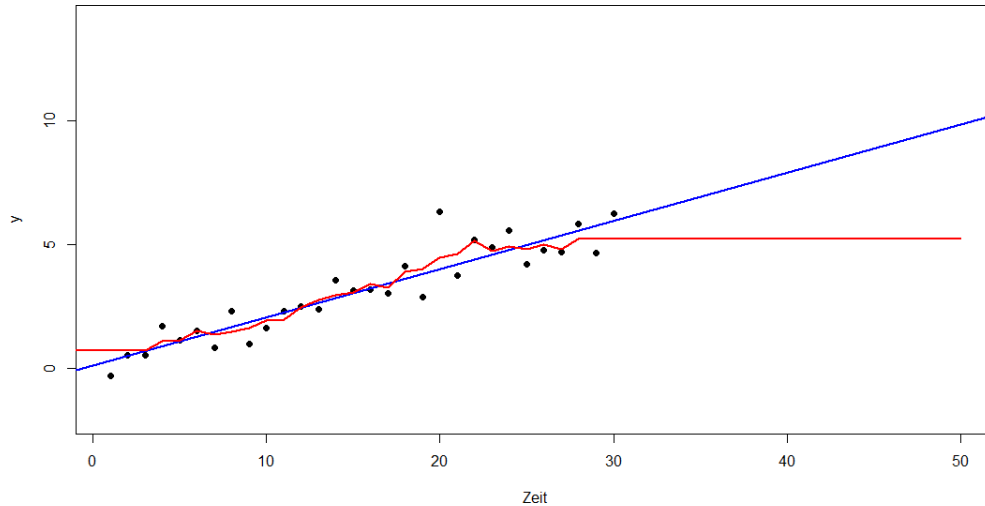
aufbereitete Nachfragedaten			
Zeit	Saison	Preis	Nachfrage
1	Winter	2.00	91
2	Winter	2.09	108
3	Winter	2.40	100
4	Winter	2.57	124
5	Winter	2.24	114
6	Winter	2.03	127
7	Winter	1.58	117
8	Fruehling	1.89	146
9	Fruehling	1.81	112
⋮	⋮	⋮	⋮
206	Winter	1.53	141
207	Winter	1.51	151
208	Winter	1.42	138

Beachten Sie hierbei, dass nun die Variable “Zeit” die durchnummerierten Wochen von 1 bis 208 enthält. Zusätzlich ist es in der Regel auch sinnvoll weitere Effekte wie Feiertage und Schulferien zu integrieren. Dies geschieht über die Bildung weiterer Spalten mit (kategorien) Variablen. Die Informationen hierzu müssen aus externen Datenquellen (z.B. Ferienkalender etc.) herangezogen werden.

Nichtlineare Effekte

Natürlich wäre es algorithmisch möglich, von einem parametrischen, linearen Modell zu einem nichtparametrischen nichtlinearen Modell überzugehen. Hierzu würde man im Beispiel der (aufbereiteten) Nachfragedaten statt der linearen Regression ein (nichtlineares) maschinelles Lernverfahren verwenden, wie zum Beispiel Support Vector Regression, k -Nearest Neighbors oder Neuronale Netze. Das ist aber nicht sinnvoll und würde in der Regel zu unsinnigen Ergebnissen wegen der Trendkomponente (im Beispiel der aufbereiteten Nachfragedaten die Variable “Zeit”) führen. Nichtparametrische Verfahren liefern grundsätzlich nur im Inneren der Datenpunkte-Wolke sinnvolle Ergebnisse, eine Extrapolation über den Rand der Punktwolke hinaus

ist nicht (oder nur in einem sehr, sehr begrenzten Rahmen möglich). Schätzt man jedoch eine Trendkomponente, um Prognosen für die Zukunft anzustellen, dann extrapoliert man immer über den Randbereich (Gegenwart) hinaus (in die Zukunft). Folgendes Beispiel zeigt eine solche Extrapolation durch **lineare Regression** (in blau) und **k-Nearest Neighbors** (in rot):



Extrapolation eines Trends mit linearer Regression (blau) und k-Nearest Neighbors (rot)

Wenn man einen U-förmigen Trend (erst fallend, dann steigend) oder einen umgekehrt-U-förmigen Trend (erst steigend, dann fallend) vermutet, kann dies durch Integration eines **quadratischen Terms** abgebildet werden. Das Modell würde im Beispiel der Nachfragedaten dann folgendermaßen aussehen:

$$\begin{aligned} \text{Nachfrage} = & \beta_0 + \beta_1 \cdot \text{Preis} + \beta_{2,1} \cdot \text{Zeit} + \beta_{2,2} \cdot \text{Zeit}^2 + \\ & + \beta_3 \cdot \text{Sommer} + \beta_4 \cdot \text{Herbst} + \beta_5 \cdot \text{Winter} + \\ & + \text{Zufallsfehler} \end{aligned}$$

Dieses Modell kann nach wie vor mit der gewöhnlichen linearen Regression berechnet werden. Hierzu muss lediglich im Datensatz eine neue Spalte “QuadrierteZeit” gebildet werden, in der die quadrierten Werte der Werte aus der Spalte “Zeit” stehen. Die neue Variable “QuadrierteZeit” wird dann zusätzlich zur Variable “Zeit” als normale Einflussvariable in der linearen Regression verwendet.

Erwartet man nichtlineare Effekte bei den anderen Einflussvariablen, dann bietet sich ein additives Modell (Unterkapitel 8.2.2) an. Hierbei muss aber darauf geachtet werden, dass die Trendkomponente als linearer Effekt berechnet wird und nichtlineare Komponentenfunktionen f_j nur bei den anderen Einflussvariablen berechnet werden. Dies ist zum Beispiel beim Verfahren L_2 -Boosting aus Unterkapitel 8.2.2 möglich. Ein geeignetes Modell im Fall der Nachfragedaten wäre zum Beispiel:

$$\begin{aligned}\text{Nachfrage} = & \beta_0 + f_1(\text{Preis}) + \beta_2 \cdot \text{Zeit} + \\ & + \beta_3 \cdot \text{Sommer} + \beta_4 \cdot \text{Herbst} + \beta_5 \cdot \text{Winter} + \\ & + \text{Zufallsfehler}\end{aligned}$$

Während die Zeit in diesem Modell als lineare Trendkomponente eingeht, kann der Preis über die (beliebig komplexe) Komponentenfunktion f_1 mit der Nachfrage verbunden sein. Die Saisonkomponenten werden nach wie vor als “lineare” Komponenten behandelt, da es sich hierbei um Dummy-Variablen für eine kategorielle Variable handelt (hierfür werden im additiven Modell stets lineare Komponenten verwendet).

10.4 Zuwachs-Modell

Vor allem wenn es in der Anwendung darum geht, aus den vorherigen Werten der Zeitreihe den nächsten (noch unbekannten) Wert vorherzusagen, dann bieten sich auch Zuwachs-Modelle an. Dabei ist im überwachten Lernproblem die Zielvariable nicht mehr der absolute Wert y_i der ursprünglichen Zielvariable, sondern die Veränderung (Zuwachs) z_i gegenüber dem vorherigen Wert:

$$z_i = y_i - y_{i-1}.$$

Wenn zum Zeitpunkt der Prognose der Wert y_{i-1} bereits bekannt ist und $\hat{z}_i$ die Prognose für die Veränderung (Zuwachs) ist, dann lautet die Prognose $\hat{y}_i$ für den nächsten Wert von y_i

$$\hat{y}_i = y_{i-1} + \hat{z}_i.$$

Gegebenenfalls ist es dann auch sinnvoll bei den Einflussvariablen ebenfalls auf die Veränderungen überzugehen, anstatt die absoluten Werte zu verwenden. Im Beispiel der Nachfrageprognose wäre dann die neue Zielvariable

$$\text{Nachfrageänderung} = \text{Nachfrage} - \text{Nachfrage der Vorwoche}$$

und die einzige Einflussvariable wäre

$$\text{Preisaenderung} = \text{Preis} - \text{Preis der Vorwoche}$$

Der entsprechend aufbereitete Datensatz sieht dann folgendermaßen aus:

Nachfrage-Rohdaten				aufbereitete Nachfragedaten		
Jahr	KW	Preis	Nachfrage	Preis-aenderung	Nachfrage-aenderung	Nachfrage
1	1	2	91	—	—	91
1	2	2.09	108	0.09	17	108
1	3	2.4	100	0.31	−8	100
1	4	2.57	124	0.17	24	124
1	5	2.24	114	−0.33	−10	114
1	6	2.03	127	−0.21	13	127
1	7	1.58	117	−0.45	−10	117
1	8	1.89	146	0.31	29	146
1	9	1.81	112	−0.08	−34	112
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Das lineare Regressionsmodell wäre

$$\text{Nachfrageaenderung} = \beta_0 + \beta_1 \cdot \text{Preisaenderung} + \text{Zufallsfehler}$$

Die Spalte “Nachfrage” wird zwar nicht in der Berechnung der Regression benötigt aber bei der Berechnung der Prognose für die Nachfrage der nächsten Woche. Die zusätzliche Integration von Saisonkomponenten ist hier nicht mehr nötig. Anstelle einer linearen Regression kann auch jedes andere überwachte Lernverfahren (z.B. Support Vector Regression, k -Nearest Neighbors, Neuronale Netze) verwendet werden.

10.5 Autoregressions-Modell

Als weitere Alternative können auch Autoregressions-Modelle verwendet werden. Die Idee dabei ist: Wenn der vergangene Wert y_{i-1} der Zielvariable einen Einfluss auf den nächsten Wert y_i der Zielvariable hat, dann kann der vergangene Wert y_{i-1} als Einflussvariable in das Modell integriert werden. Ein entsprechendes Regressionsmodell in unserem Nachfrage-Beispiel würde dann lauten:

$$\text{Nachfrage} = \beta_0 + \beta_1 \cdot \text{NachfrageVorwoche} + \beta_2 \cdot \text{Preis} + \text{Zufallsfehler}$$

Beachten Sie, dass dabei die Nachfrage der Vorwoche (“NachfrageVorwoche”) nicht eins zu eins in die Prognose der kommenden Nachfrage eingeht, sondern nur mit einem Faktor β_1 (dem Regressionskoeffizienten), der aus den Daten errechnet wird. Wenn die Höhe der Nachfrage in der Vorwoche kaum einen Einfluss auf die aktuelle Nachfrage hat, dann wird sich bei der Berechnung der Regressionsergebnisse für den Koeffizienten β_1 auch nur ein sehr kleiner Wert (nahe 0) ergeben.

Der entsprechend aufbereitete Datensatz sieht dann folgendermaßen aus:

Nachfrage-Rohdaten				aufbereitete Nachfragedaten		
Jahr	KW	Preis	Nachfrage	Preis	NachfrageVorwoche	Nachfrage
1	1	2.00	91	2.00	–	91
1	2	2.09	108	2.09	91	108
1	3	2.40	100	2.40	108	100
1	4	2.57	124	2.57	100	124
1	5	2.24	114	2.24	124	114
1	6	2.03	127	2.03	114	127
1	7	1.58	117	1.58	127	117
1	8	1.89	146	1.89	117	146
1	9	1.81	112	1.81	146	112
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Das Autokorrelations-Modell kann auch dahingehend erweitert werden, dass nicht nur die Nachfrage der letzten Woche als Einflussvariable verwendet wird, sondern zusätzlich auch die Nachfrage aus der vorletzten Woche, der vorvorletzten Woche etc. Die zusätzliche Integration eines Trendes oder von Saisonkomponenten ist hier nicht unbedingt nötig, da diese Effekte bereits in der Nachfrage der Vorwoche (weitgehend) enthalten sind.

Anstelle einer linearen Regression kann auch jedes andere überwachte Lernverfahren (z.B. Support Vector Regression, k -Nearest Neighbors, Neuronale Netze) verwendet werden.

10.6 Testdatensatz bei Zeitreihen

Die Aufteilung in Trainings- und Testdatensatz sowie die Berechnung einer realistischen Prognosegüte benötigt bei Zeitreihen eine besondere Sorgfalt. Sehr häufig passieren hier Fehler, die zu völlig falschen Ergebnissen führen.

Um eine ehrliche Bewertung der Prognosegüte zu erhalten, ist es wichtig, dass bei der Berechnung der Prognosegüte exakt so verfahren wird, wie bei der realen Anwendung. Betrachten wir zum Beispiel den typischen Fall beim Online Learning: Zu jedem Zeitpunkt muss der nächste Wert y_i der Zielvariable basierend auf den vorherigen Werten $y_1, y_2, \dots, y_{i-1}$ (und weiterer Einflussvariablen) prognostiziert werden und das maschinelle Lernverfahren wird vor jeder Prognose anhand der größeren Datenbasis neu trainiert. Wenn beispielsweise ein Datensatz mit 1000 Datenpunkten vorliegt, bietet sich folgendes Vorgehen an:

1. Zunächst wird das maschinelle Lernverfahren anhand der ersten 700 Datenpunkte $(x_1, y_1), \dots, (x_{700}, y_{700})$ trainiert – einschließlich Einstellung der Tuningparameter mittels Kreuzvalidierung. Der so trainierte Algorithmus prognostiziert dann lediglich den Wert des nächsten Wertes y_{701} . Die Güte dieser Prognose wird abgespeichert.
 2. Anschließend wird das maschinelle Lernverfahren anhand der ersten 701 Datenpunkte $(x_1, y_1), \dots, (x_{700}, y_{700}), (x_{701}, y_{701})$ neu trainiert. Der so trainierte Algorithmus prognostiziert dann lediglich den Wert des nächsten Wertes y_{702} . Die Güte dieser Prognose wird abgespeichert.
 - $\vdots$ $\vdots$
 300. Anschließend wird das maschinelle Lernverfahren anhand der ersten 999 Datenpunkte $(x_1, y_1), \dots, (x_{700}, y_{700}), (x_{701}, y_{701}), \dots, (x_{999}, y_{999})$ neu trainiert. Der so trainierte Algorithmus prognostiziert dann lediglich den Wert des nächsten Wertes y_{1000} . Die Güte dieser Prognose wird abgespeichert.
- Auf diese Weise erhält man 300 Werte für die Prognosegüte. Der Durchschnitt dieser 300 Werte ist dann der ermittelte Endwert der Prognosegüte des Algorithmus.

Je nachdem in welchen Abständen die Werte der Tuningparameter mittels Kreuzvalidierung neu justiert werden, muss dies auch bei der Berechnung der Prognosegüte nachgebildet werden. Bei der Berechnung der Prognosegüte soll also der tatsächliche Einsatz in der Praxis so originalgetreu wie möglich nachgebildet werden.

10.7 Zusammenfassung und ergänzende Hinweise

In den vorherigen Unterkapiteln wurden Methoden für überwachte Lernprobleme bei Zeitreihen besprochen. Die hier besprochenen Methoden sind keine eigenständigen Lernverfahren, sondern es sind Methoden der Modellierung. Modellierung bedeutet hierbei: die Bildung weiterer Einflussvariablen aus den ursprünglichen Daten. Es lässt sich nicht pauschal beantworten, welche der in den verschiedenen Unterkapiteln vorgestellten Verfahren in welcher Praxis-situation am besten sind. Die hier vorgestellten Methoden bilden eher einen Werkzeugkasten für die Modellbildung und keine Blaupausen für fertige Modelle. Die Wahl eines geeigneten Modells hängt von den konkreten Gegebenheiten im Praxisfall ab. Im Zuwachs-Modell und im Autokorrelations-Modell basieren Prognosen für die Zielvariable y_i zum Zeitpunkt i entscheidend auf dem letzten Wert y_{i-1} der Zielvariable. Falls aber in der konkreten Praxisanwendung dieser Wert nicht bekannt ist, dann können die Modelle so nicht eins zu eins übernommen werden. Dies ist zum Beispiel der Fall, wenn nicht nur der nächste Folgewert einer Zeitreihe prognostiziert werden soll, sondern wenn Prognosen auch für eine etwas fernere Zukunft berechnet werden sollen. In diesem Fall bietet sich evtl. eher ein Modell mit Trend und Saisonkomponenten an. Möglicherweise ist zu dem Zeitpunkt, an dem die Prognose für y_i berechnet werden muss, zwar der Wert y_{i-1} nicht bekannt, aber zumindest bereits der Wert y_{i-2} . Dann könnten entsprechende Modifikationen von Zuwachs- und Autoregressionsmodellen verwendet werden.

Wenn jeweils zu einem bestimmten Zeitpunkt der nächste (oder übernächste o.Ä.) Wert der Zielvariable prognostiziert werden soll (Online Learning), dann kann das Modell vor jeder Prognose neu trainiert werden. Im Lauf der Zeit kommen immer neue Datenpunkte hinzu und durch das kontinuierliche Trainieren des Algorithmus verbessert sich der Algorithmus selbstständig im Lauf der Zeit. Häufig ist es auch ratsam alte Datenpunkte irgendwann aus den Trainingsdaten zu entfernen, so dass das Training eher auf aktuellen Daten beruht und der Algorithmus sich schneller an grundlegende Veränderungen, die im Lauf der Zeit eintreten können, anpasst.

Falls die Abstände zwischen den Prognosezeitpunkten kurz sind (beispielsweise nur eine Sekunde statt eine Woche wie in unserem Nachfragebeispiel), ist es evtl. nicht möglich, den Algorithmus laufend in Echtzeit zu trainieren. In diesem Fall kann das Trainieren des Algorithmus auf einem separaten Server erfolgen und ein Update der Prognosefunktion erfolgt nur in regelmäßigen Abständen. Um Rechenzeit zu sparen, kann auch darauf verzichtet werden,

in jedem Trainingsschritt eine neue Einstellung der Tuningparameter (etwa durch Kreuzvalidierung) zu berechnen. Speziell wenn das Verfahren vor jeder Prognose neu trainiert wird, ändert sich die Datenbasis jeweils nur minimal, so dass eine ständige Anpassung der Tuningparameter in der Regel unnötig ist. Es reicht, wenn eine Neujustierung der Tuningparameter von Zeit zu Zeit in regelmäßigen Abständen stattfindet.

Vor allem bei Zeitreihen ist der Prozess der Modellbildung immer auch ein bisschen “Basterei”. Glücklicherweise lässt sich die Güte des Modells (und damit die Nähe zur Realität) in überwachten Lernproblemen stets über die Prognosegüte sauber validieren. Beim Prozess der Modellbildung muss aber folgender Leitsatz immer beachtet werden:

“Es können nur diejenigen Daten als Einflussvariablen gewählt werden, die zu dem Zeitpunkt, an dem die Prognose berechnet werden muss, auch tatsächlich vorliegen!”

Zusätzlich zu den hier besprochenen Methoden gibt es auch Maschinelle lernverfahren, die speziell für Zeitreihen konzipiert wurden. Das populärste dieser Verfahren ist LSTM (Long Short-term Memory). Hierbei handelt es sich um ein Neuronales Netz, das speziell für sequenzielle Daten (wie z.B. Zeitreihen) entwickelt wurde. LSTM sind zum Beispiel im R-Paket **TSLSTM** implementiert:

<https://cran.r-project.org/web/packages/TSLSTM/index.html>

Von Facebook wurde für Prognosen bei Zeitreihen das R-Paket **prophet** entwickelt:

<https://cran.r-project.org/web/packages/prophet/index.html>

Es basiert (wie das L_2 -Boosting) auf additiven Modellen und ist speziell für Daten mit starken saisonalen Effekten sehr gut geeignet.

11 Unsupervised Learning

11.1 Einführung

Bei einer Regression oder einer Klassifikation ist es das Ziel den Zusammenhang zwischen Einflussgrößen und einer Zielgröße zu schätzen:

$$\text{Einflussvariablen} \longrightarrow \text{Zielvariable}$$

Hierzu werden sowohl Daten für die Einflussgrößen als auch für die Zielgrößen benötigt.

Häufiges Problem: Oftmals liegen zwar Daten für die Einflussgrößen vor, aber keine Daten für die Zielgröße. Oder es liegen zwar Daten vor, aber die Daten für die Einfluss- und Zielgrößen wurden getrennt erhoben und lassen sich nicht sauber Zusammenführen. (Beispiel: Qualität in Abhängigkeit von Maschineneinstellungen in der Produktion)

Weil bei einer Regression oder Klassifikation sowohl Daten für die Einflussgrößen als auch für die Zielgrößen vorliegen, handelt es sich hierbei um sogenannte **überwachte Lernprobleme** (supervised learning). Wenn keine Zielgröße existiert (oder hierfür keine Daten vorliegen) handelt es sich um ein **unüberwachtes Lernproblem** (unsupervised learning).

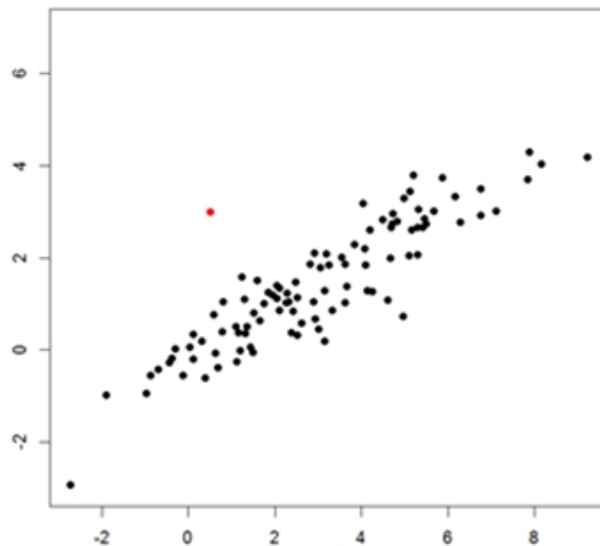
- überwachte Lernprobleme (supervised learning)
 - Regression
 - Klassifikation
- unüberwachte Lernprobleme
 - Anomalieerkennung
 - Clusteranalyse

Bei unüberwachten Lernproblemen geht es nicht darum, Zusammenhänge zwischen Einfluss- und Zielgrößen zu finden, sondern die Daten zu strukturieren. Also zum Beispiel die Daten in normale und anormale Daten aufzuteilen oder Gruppen von Daten zu identifizieren.

Beispiel Anomalieerkennung

Anomalieerkennung wird zum Beispiel verwendet

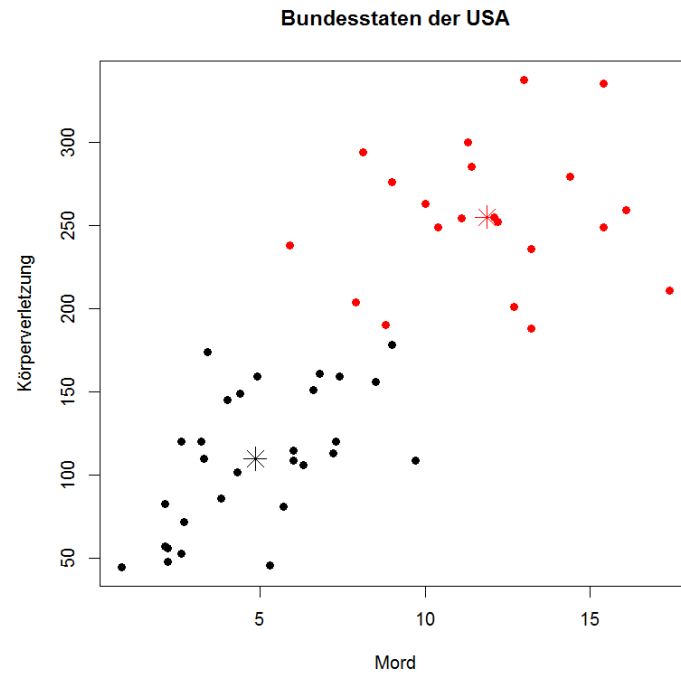
- zur Erkennung von Betrugsfällen in Buchungsdaten
- bei automatischen Warnsystemen



Beispiel Clusteranalyse

In der Clusteranalyse werden Gruppen von Daten (sog. Cluster) identifiziert. Dies wird häufig im Marketing eingesetzt, um zum Beispiel die Kunden in verschiedene Kundengruppen aufzuteilen. Die verschiedenen Kundengruppen können dann zum Beispiel mit unterschiedlichen Marketingaktionen adressiert werden.

Ein anderes Beispiel für eine Clusteranalyse sehen Sie in nachfolgendem Bild. Darin sind alle Bundesstaaten der USA als Punkte eingetragen. Auf der horizontalen Achse ist die Anzahl an Morde, auf der vertikalen Achse die Anzahl der Körperverletzungen (relativ zur Einwohnerzahl) aufgetragen. Man sieht, dass die Zahlen für Morde und Körperverletzungen relativ stark korrelieren. Außerdem lassen sich die Bundesstaaten recht gut in zwei Gruppen einteilen: solche mit relativ wenigen Fällen von Morden und Körperverletzungen (schwarze Punkte) und solche mit relativ vielen Fällen von Morden und Körperverletzungen (rote Punkte).



11.2 Clusteranalyse mit k -Means

Worum geht es?

Das k -Means-Verfahren ist ein mathematisches Verfahren zur Aufteilung von Datenpunkten in Gruppen (Cluster). Das k in k -Means steht dabei für die Anzahl der Gruppen.

Was ist die Voraussetzung?

Für das Verfahren sollten nur die Spalten des Datensatzes verwendet werden, die metrische Daten (keine kategorielle Daten) enthalten.

Wie ist die Funktionsweise des Verfahrens?

1. Manuelle Wahl der Anzahl k der Gruppen (Cluster)
2. Jede Guppe (Cluster) ist durch ein Zentrum definiert.
3. Jeder Datenpunkt gehört zu dem Zentrum (d.h. der Gruppe/Cluster), das ihm am nächsten liegt.
4. Das Verfahren wählt die Zentren so, dass die Summe der Abstände zwischen den Datenpunkten und ihren jeweiligen Zentren minimal sind.

Achtung: Die so gebildeten Gruppen (Cluster) sind (i.d.R.) nicht gleich groß: unterschiedliche Anzahl an Datenpunkten je Gruppe, unterschiedliche (räumliche) Ausdehnung des Gebiets der Gruppe.

Wenn Sie ein Gefühl für die Funktionsweise des k -Means-Algorithmus bekommen wollen, kann ich Ihnen wärmstens die Internetseite

<https://www.naftaliharris.com/blog/visualizing-k-means-clustering>

empfehlen. Die Seite bietet eine interaktive Visualisierung und ermöglicht Ihnen, ein wenig mit dem Verfahren zu spielen.

11.3 Anomalieerkennung mit One-Class-SVM

Zur Anomalieerkennung gibt es sehr viele verschiedene Lernalgorithmen. Ein sehr populärer Algorithmus ist zum Beispiel Isolation Forest. Hierbei handelt es sich um eine Abwandlung des eigentlich überwachten Lernverfahrens Random Forest, das wiederum auf Entscheidungsbäumen beruht.

Eine weiteres populäres Verfahren zur Anomalieerkennung ist die One-Class-SVM. Diese ist im Prinzip wie eine ganz normale SVM zur Klassifikation aufgebaut:

$$\min_{f \in H} \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)) + \underbrace{\lambda \cdot \text{Komplexität}(f)}_{\text{Regularisierung}}$$

Da wir es hier mit einem unüberwachten Lernproblem zu tun haben, gibt es eigentlich gar keine Werte/Daten für y_i in dieser Formel. Die Idee ist nun aber, dass zum Trainieren der SVM für alle Datenpunkte x_i der Wert $y_i = 1$ gesetzt wird. Daher heißt es auch One-Class-SVM, da die SVM nur mit Werten aus einer Klasse (eben der Klasse $y = 1$) trainiert wird. Die Trainingsdaten für die SVM sind also

$$(x_1, 1), (x_2, 1), (x_3, 1), \dots, (x_n, 1).$$

Einen Testdatensatz gibt es nicht, da wir ja im unüberwachten Lernen sind.

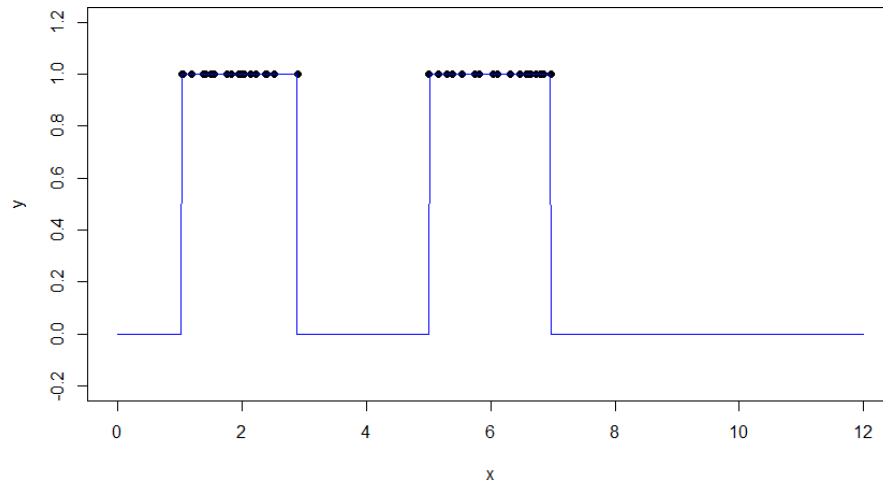
Dadurch dass alle Daten $x_1, x_2, \dots, x_n$ aus dem Datensatz künstlich der Klasse $y = 1$ zugeordnet werden, lernt die SVM, dass die Daten aus dem Ursprungsdatensatz (normalerweise) in Klasse $y = 1$ liegen. Auch Datenpunkte, die den Daten aus dem Ursprungsdatensatz sehr ähnlich sind, werden von der SVM dann auch der Klasse $y = 1$ zugeordnet. Unterscheidet sich ein Datenpunkt jedoch deutlich von den Trainingsdaten (Anomalie), dann wird der Datenpunkt der Klasse $y = -1$ zugeordnet.

Auf den ersten Blick wirkt es äußerst erstaunlich, dass dies funktioniert, denn der so verwendete Trainingsdatensatz ist ja extrem imbalanced. Man könnte daher vermuten, dass die aus den Trainingsdaten errechnete Prognosefunktion konstant gleich 1 ist

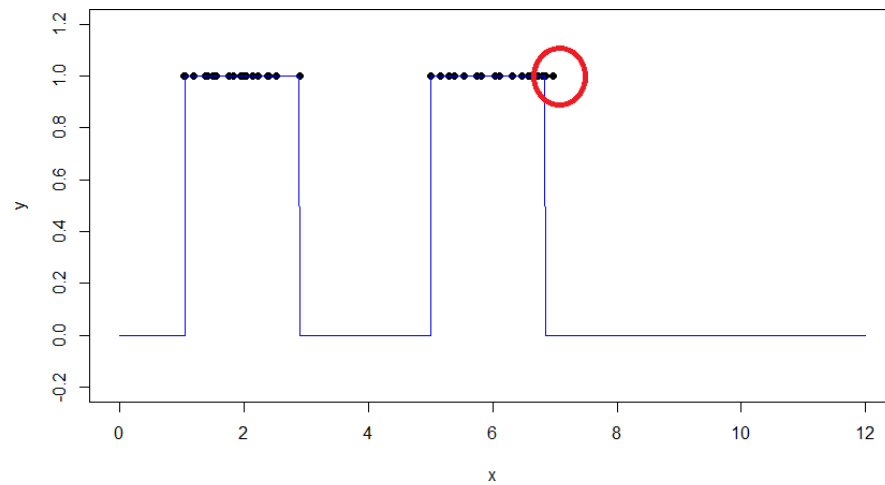
$$\hat{f} \equiv 1$$

und somit für jedes x immer auf $y = \hat{f}(x) = 1$ entscheidet. Aufgrund der sehr speziellen Konstruktionsweise der SVM-Funktionen f im Funktionenraum H und der speziellen Messung der Komplexität passiert dies jedoch nicht, denn

der Funktion $\hat{f} \equiv 1$ wird von der SVM in der Tat eine sehr hohe Komplexität zugeordnet! Die SVM tendiert in ihrer Komfortzone dazu auf 0 zurück zu schwingen, wie folgendes kleine Beispiel mit 40 Datenpunkten zeigt:

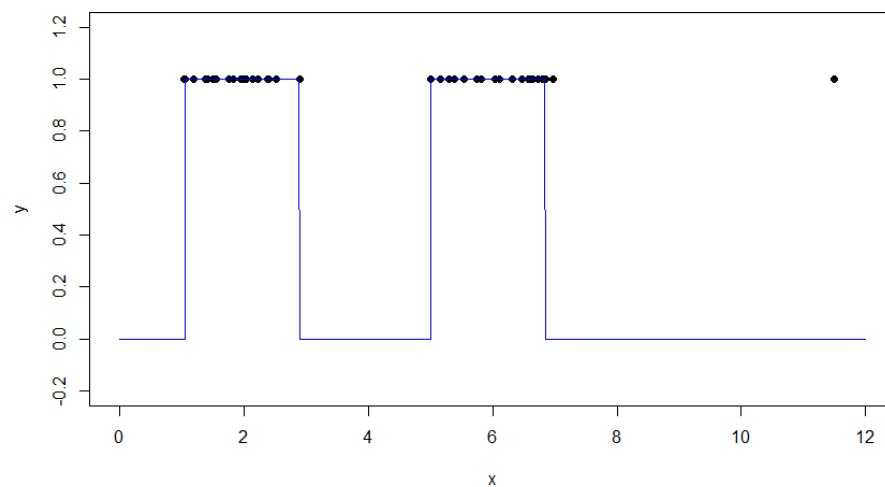


Ein Vorteil der One-Class-SVM ist, dass der Trainingsdatensatz nicht frei von Anomalien sein muss. Zwar lernt die One-Class-SVM aus den Trainingsdaten, was “normale” Datenpunkte sind, doch darf im Trainingsdatensatz auch ein gewisser Prozentsatz von Anomalien enthalten sein. Dieser Prozentsatz kann über den Tuningparameter ν gesteuert werden. Der Tuningparameter ν gibt die Obergrenze für den Anteil an Anomalien an und entspricht dem Regularisierungsparameter λ , der die Komplexität der SVM steuert. Im Beispiel von gerade eben war ν so klein eingestellt, dass keiner der 40 Trainingsdaten als Anomalie eingestuft wurde. In folgender Grafik wurde ν erhöht auf $\nu = 0.1$, so dass bis zu 10% der Trainingsdaten als Anomalie eingestuft werden können:



Man sieht, dass es am rechten Ende (rot umkreister Bereich) einen Datenpunkt gibt, der als Anomalie eingestuft wird, da die blaue Linie schon vor dem Datenpunkt nach unten wegbreicht. Für die Wahl von ν muss man sich bewußt machen, dass – je nach Einstellung von ν – die SVM erzwungenermaßen auch dann Anomalien im Datensatz “findet”, wenn keine vorhanden sind.

Für folgende Grafik wurde als 41. Datenpunkt eine echte Anomalie (Ausreißer ganz rechts bei $x = 11.5$) eingefügt; der Tuningparameter ν ist unverändert $\nu = 0.1$. Die Anomalie wird von der SVM erkannt:



Validierung von Ergebnissen von Anomalie-Erkennungsalgorithmen:

In unserem kleinen Beispiel konnten wir die Ergebnisse der Anomalieerkennung visuell nachzuvollziehen. Dies ist aber nur bei bis zu zwei Variablen möglich. Bei z.B. $m = 10$ Variablen müssten wir zur visuellen Validierung eine Grafik im $\mathbb{R}^{10}$ zeichnen, was sich als schwierig erweisen dürfte ...

Wie immer beim unüberwachten Lernen gibt es keine befriedigende Lösung die Ergebnisse zu validieren (und auch Tuningparameter wie z.B. ν fundiert einzustellen).

In der Praxis ist es daher ratsam auf sogenanntes semi-supervised learning überzugehen, auch wenn dies mit einem erheblichen manuellen Aufwand verbunden ist: Der Algorithmus zur Anomalieerkennung (z.B. die One-Class-SVM) erhält zum Trainieren einen unüberwachten Datensatz (ohne Labels y). Zusätzlich wird jedoch ein überwachter Testdatensatz gebildet, in dem ein Experte manuell Angaben gemacht hat, welche Datenpunkte normal ($y = 1$) sind, und bei welchen Datenpunkten es sich um Anomalien ($y = -1$) handelt. Anschließend kann der auf dem Trainingsdatensatz trainierte Algorithmus auf dem Testdatensatz angewendet werden. Durch die vorhandenen Labels y auf dem Testdatensatz können dann Gütemaße wie bei einer Klassifikation berechnet werden, die die Leistungsfähigkeit des trainierten Anomalie-Erkennungsalgorithmus messen.

Dieses Vorgehen bedeutet zwar in der Regel einen erheblichen manuellen Aufwand, sollte jedoch – wann immer möglich (!) – unbedingt angewendet werden, da sich die Anomalieerkennung ansonsten im völligen Blindflug bewegt. Auch wenn aufgrund der praktischen Gegebenheiten nur ein sehr kleiner gelabelter Testdatensatz möglich ist, so ist dies immer noch um Welten besser als gar nichts!