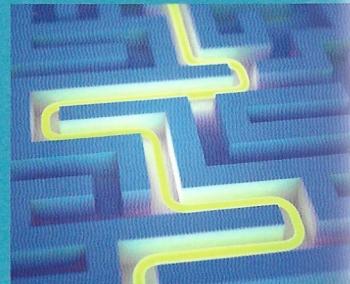
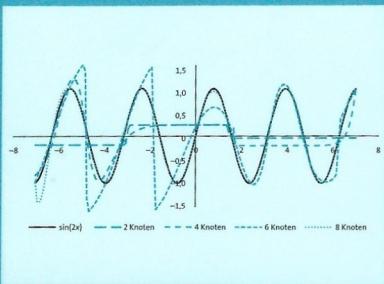




# GRUNDKURS

# Machine Learning



- + Mathematische Grundlagen des maschinellen Lernens
- + Alle wichtigen Algorithmen Schritt für Schritt erklärt
- + Inkl. Reinforcement Learning, k-nächste Nachbarn, Neuronale Netze u. v. m.

## Prinzipielle Kategorien

### ► Überwachtes Lernen

Beim *überwachten Lernen* sind Eingabe und Ausgabe gegeben. Die Eingabe liegt als Vektor vor, d. h. mit typischerweise mehr als einer Variablen. Nehmen wir z. B. digitale Bilder von Hunden, die zu Vektoren transformiert werden, die die Pixelfarben repräsentieren. Diese Bilder wurden als Jagdhund, Pudel, Terrier etc. klassifiziert. Wir trainieren nun unseren Algorithmus auf Basis dieser Bilder in der Hoffnung, dass ein neues Bild korrekt klassifiziert wird. Natürlich können wir überwachtes Lernen auch für Regressionsaufgaben verwenden, in denen für jede Eingabe numerische Werte für die jeweilige Ausgabe vorhergesagt werden (zum Beispiel besteht die Eingabe aus Temperatur und Tageszeit, und wir versuchen, den Stromverbrauch vorherzusagen). Überwachtes Lernen umfasst viele verschiedene Algorithmen, von denen wir einige im Folgenden erwähnen.

### ► Unüberwachtes Lernen

Man spricht von *unüberwachtem Lernen*, wenn die Eingabedaten beim Training nicht mit passenden Ausgabewerten versehen sind. Das bedeutet, wir haben nur Eingabedaten, keine Ausgabedaten (die auch zuweilen als *Label* oder *Kennzeichnung* bezeichnet werden und die Daten als *gekennzeichnet*). Der Algorithmus ist gewissermaßen auf sich allein gestellt. Dieser Algorithmus hat eher die Aufgabe, Beziehungen oder Muster für Sie zu finden. Wir zeigen dem Computer digitale Hundebilder, und dieser versucht, diese Bilder zu gruppieren oder zu bündeln auf Basis irgendwelcher Eigenschaften, die *er* für am wichtigsten hält. Möglicherweise erhalten wir als Ergebnis Gruppen oder Cluster, die wir dann als Jagdhund, Pudel oder Terrier ... oder vielleicht als schwarz, braun, weiß ... bezeichnen. Oder der Computer liefert Gruppen nach vollkommen unterschiedlichen Kriterien, die wir Menschen nicht mal bemerken würden. Unüberwachtes Lernen mag auf den ersten Blick etwas eigenartig erscheinen, aber um es zu verstehen, können Sie sich vorstellen, ich würde zu Ihnen sagen: »Hier ist der Inhalt meiner Schreibtischlade. Bitte sortiere die Sachen.« Auch für das unüberwachte Lernen gibt es viele verschiedene Algorithmen, die wir uns später noch genauer ansehen.

### ► Verstärkendes Lernen

Beim *verstärkenden Lernen* lernt ein Algorithmus, etwas zu *tun*. Dabei wird der Algorithmus für erfolgreiches Vorgehen belohnt und/oder für misslungenes Verhalten bestraft. (Womit wir wieder bei den Hunden wären!) Das zuvor erwähnte MENACE war dafür ein einfaches Beispiel, Belohnungen und Bestrafungen bestanden dabei aus Zugabe oder Wegnahme von Streichhölzern zu oder von den Schachteln. Computer erfolgreich Brettspiele oder Videospiele aus den 80ern spielen zu lassen, scheint

## Support-Vektor-Maschinen

Support-Vektor-Maschinen sind eine Klassifikationstechnik des überwachten Lernens. Die klassifizierten Daten werden durch Merkmalsvektoren repräsentiert. Diese Methode unterteilt Daten je nach dem, auf welcher Seite einer Hyperebene ein Datenpunkt im Merkmalsraum liegt.

### 7.1 Wofür können wir sie verwenden?

Support-Vektor-Maschinen werden verwendet, um Daten anhand numerischer Merkmale zu klassifizieren. Beispiele sind:

- Pflanzentypen aufgrund von Merkmalen ihrer Blütenblätter identifizieren
- Prostatakrebsrisiko anhand von MRT-Bildern einschätzen
- Potenzielle Kunden anhand ihrer Twittereinträge aufspüren

Die Support-Vektor-Maschine (SVM) ist eine weitere Technik der überwachten Klassifikation. Wir brauchen dazu ein bisschen mehr Mathematik im Vergleich zu den bisherigen Methoden, deshalb habe ich sie mir bis hierhin aufgespart.

Man kann sie immer noch mit Excel umsetzen, zumindest, was das Lernen betrifft – unter der Voraussetzung, dass man nicht zu viele Daten hat. Für alle hier diskutierten Methoden gilt: Für den professionellen Einsatz müssen sie richtig und ordentlich programmiert werden.

### 7.2 Harte Ränder

Im Abbildung 7.1 zeige ich einige Vektoren, die in zwei Klassen eingeteilt sind: Kreise und Rauten. Es gibt ganz klar zwei Gruppen. Diese beiden Klassen können durch eine Gerade getrennt werden, sie sind also linear separierbar.

Diese Trennerade hilft uns zu bestimmen, in welche Klasse ein neuer Datenpunkt gehört, abhängig davon, auf welcher Seite der Geraden sich der Punkt befindet. Mit so einer

klaren Abgrenzung der Gruppen sollte es ein Leichtes sein, eine solche Gerade zu finden. Es ist allerdings sogar so einfach, dass es viele mögliche Trenngeraden gibt, wie in der Abbildung zu sehen ist. Welche ist nun die beste?

Eine mögliche Definition einer besten Geraden ist jene, die zur breitesten Trennfläche (bzw. zu Rändern mit größtem Abstand) zwischen den Datenpunkten führt. Diese Gerade wäre die fett gedruckte Linie in Abbildung 7.1.

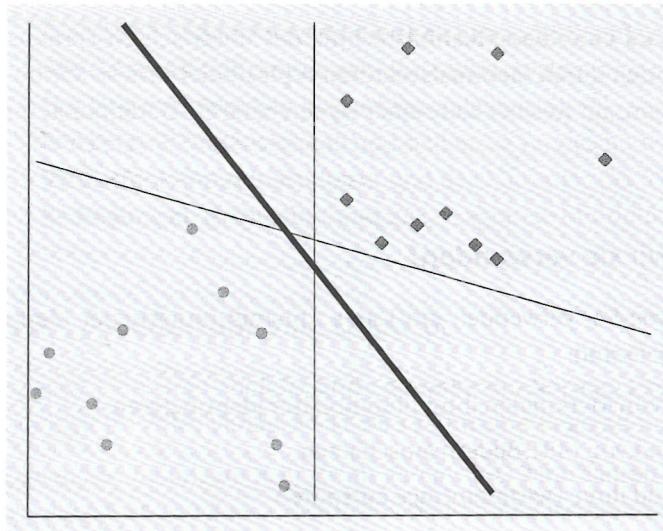


Abbildung 7.1 Zwei Klassen und mögliche Trennlinien

Die Ränder sind in Abbildung 7.2 zu sehen, wo ich auch einen Vektor eingezeichnet habe, der orthogonal zur klassentrennenden Hyperebene liegt. Ich sage absichtlich Hyperebene, da SVM üblicherweise für mehrdimensionale Aufgaben eingesetzt wird, hier ist die Hyperebene aber bloß eine Gerade.

Unser Ziel ist es, die Hyperebene zu finden, die zu Rändern mit größtmöglichem Abstand führt. Die Ränder liegen auf sogenannten Support-Vektoren (manchmal auch als Stützvektoren bezeichnet).

Dieser Abschnitt heißt »Harte Ränder«. Das soll darauf hinweisen, dass es eine klare Abgrenzung zwischen den beiden Klassen gibt. Wenn auch nur ein Datenpunkt auf der falschen Seite liegt, zum Beispiel eine Raute auf der Kreisseite, dann müssen wir uns dazu etwas anderes überlegen. Das werden wir uns in Abschnitt 7.5, »Weiche Ränder«, und Abschnitt 7.6, »Kernel-Trick«, genauer ansehen.

# Kapitel 9

## Entscheidungsbäume

**Entscheidungsbäume** gehören zu den Methoden des überwachten Lernens. Eigentlich sind sie nichts anderes als Flussdiagramme. Sie können sowohl für die Klassifikation als auch für die Regression verwendet werden, deshalb werden sie auch manchmal so bezeichnet: *Klassifikations- und Regressionsbäume*, auch unter der Abkürzung *CART* bekannt. Nutzen wir sie für die Klassifikation, dann starten wir wieder mit einem annotierten Datensatz mit mehreren Merkmalen. Diese Merkmale beschreiben jeden Datenpunkt entsprechend ihrer Attribute (zum Beispiel groß oder klein, männlich oder weiblich oder auch numerische Werte). Die Methode basiert auf einer ständigen Unterteilung der Daten abhängig von ihren Attributen. Diese Bäume können auch für Regressionsaufgaben verwendet werden, wenn die Daten statt mit Klassen mit numerischen Werten assoziiert werden.

9

### 9.1 Wofür können wir sie verwenden?

Entscheidungsbäume werden verwendet, um Daten nach kategorischen oder numerischen Attributen zu klassifizieren. Beispiele dafür sind:

- ▶ Vorhersage der Punkte beim Eurovision Song Contest durch Betrachtung des Liedtyps oder der Beats pro Minute, der Anzahl an Bandmitgliedern etc.
- ▶ Prognose des IQ basierend auf gelesenen Büchern, Buchtyp, -themen, -autoren etc.

Ohne Zweifel haben Sie schon einmal Entscheidungsbäume gesehen, aber vielleicht nicht im Zusammenhang mit maschinellem Lernen. Sie haben sicher schon einmal *Personenraten* gespielt. »Ist der Schauspieler männlich?« Ja. »Ist er unter 50?« Nein. »Hat er eine Glatze?« Ja. »Bruce Willis?« Ja. Wäre die Antwort »Nein«, dann hätten Sie einen anderen Weg durch den Baum der Möglichkeiten genommen.

Beim Personenraten besteht der Trick darin, in jeder Phase die besten Ja/Nein-Fragen zu finden, sodass man auf dem schnellstmöglichen Weg zur Antwort kommt. Maschinelles Lernen verfolgt ein ähnliches Ziel. Sie starten mit einem klassifizierten Trainingsdatensatz und nutzen diesen, um die bestmögliche Baumstruktur aus Fragen und Antworten zu konstruieren, um damit bei einem neuen Datenpunkt möglichst schnell und genau

zu einer Klassifikation zu kommen. Also gehören Entscheidungsbäume zu den wachten Lernverfahren. Die Fragen im Entscheidungsbaum müssen nicht binär sein, und die Antworten können auch Zahlen sein.

Ich kann Ihnen nicht empfehlen, Excel zu verwenden, wenn Sie mit Entscheidungsbäumen und großen Datenmengen arbeiten. Es wird einfach zu unübersichtlich. Der Grund liegt darin, dass man die Baumstruktur im Vorhinein nicht kennt, was es schwierig macht, sie in einer Tabellenkalkulation anzulegen.

Zuerst ein bisschen Jargon und Konventionen: Ein Baum wird verkehrt herum genutzt, also mit der Wurzel, der erste Frage, nach oben. Jede Frage ist eine Bedingung, ein interner Knoten, der die Merkmale bzw. Attribute teilt. Von jedem Knoten gehen ab, die die möglichen Antworten repräsentieren. Wenn Sie den Baum bis zum Ende durchlaufen, sodass es keine Fragen/Entscheidungen mehr gibt, dann befinden Sie sich auf einem Blatt. Es gibt außerdem die – naheliegende – Terminologie mit Mutter/Kinder und Kindern in Analogie zu Stammbäumen (Siehe Abbildung 9.1).

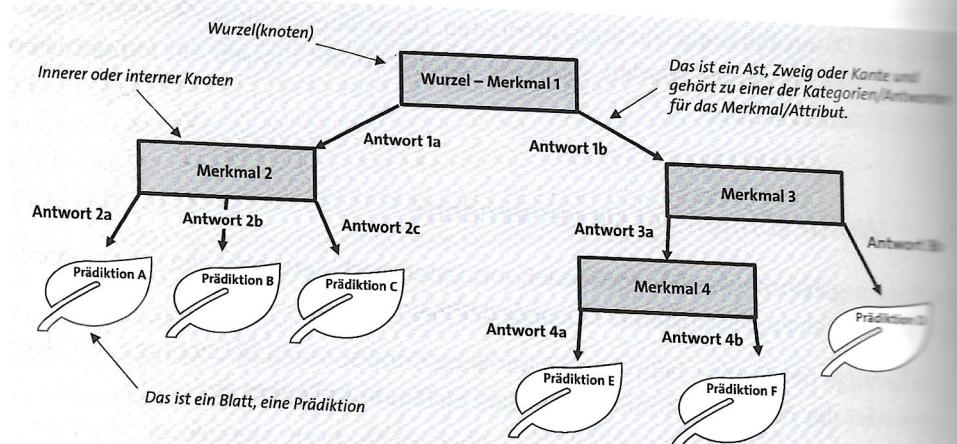


Abbildung 9.1 Ein schematisches Diagramm mit den Charakteristiken eines Entscheidungsbäumes

Sie können Entscheidungsbäume verwenden, um Daten zu klassifizieren, zum Beispiel ob ein Pilz essbar ist oder nicht aufgrund unterschiedlicher physikalischer Eigenschaften. Das können binäre Kategorien sein, wie mit oder ohne Lamellen, oder Mehrfachkategorien, wie Farbe, oder eben numerische, wie die Größe des Pilzes. Entscheidungsbäume können aber auch für die Regression eingesetzt werden, wenn Sie numerische Daten haben, wie zum Beispiel: Was ist der Wert eines Autos, basierend auf Typ, Modell etc.?

# Kapitel 10

## Neuronale Netze

*Neuronale Netzwerke* oder *Netze (NN)* gehören zu den Verfahren des maschinellen Lernens, die sich die Arbeitsweise des menschlichen Gehirns als Vorbild nehmen. Neuronen erhalten Signale, die dann mathematisch manipuliert an andere Neuronen weitergeleitet werden.

Das Eingangssignal kann mehrere Schichten an Neuronen durchlaufen, bevor es als Ausgabe in Form einer Klassifikation oder Regression herauskommt. Es kann sowohl für überwachte als auch für unüberwachte Aufgaben eingesetzt werden.

10

### 10.1 Wofür können wir sie verwenden?

Neuronale Netze werden für die Modellierung komplexer Beziehungen zwischen Eingabe und Ausgabe verwendet. Beispiele dafür sind:

- ▶ Bilderkennung, wie zum Beispiel Handschriftenerkennung
- ▶ Verbesserung körniger Bilder
- ▶ Übersetzung verschiedener Sprachen
- ▶ Werbung. Wenn Marketingleute neuronale Netze einsetzen können, dann werden sie das auch tun!

### 10.2 Ein sehr einfaches Netzwerk

Sie werden sicher schon Bilder wie in Abbildung 10.1 gesehen haben, die ein typisches neuronales Netz darstellen. Das illustriert die Struktur eines sehr einfachen vorwärtsgerichteten (*Feedforward*) neuronalen Netzes. Eingaben kommen von links, und Ausgaben gehen rechts hinaus. Dazwischen werden in jedem Knoten die Daten mathematisch manipuliert.

Man bezeichnet es deshalb als *vorwärtsgerichtet*, weil Daten und Berechnungen nur in eine Richtung gehen, es gibt also keine Rückkopplung, wie Sie sie später noch kennenlernen werden.

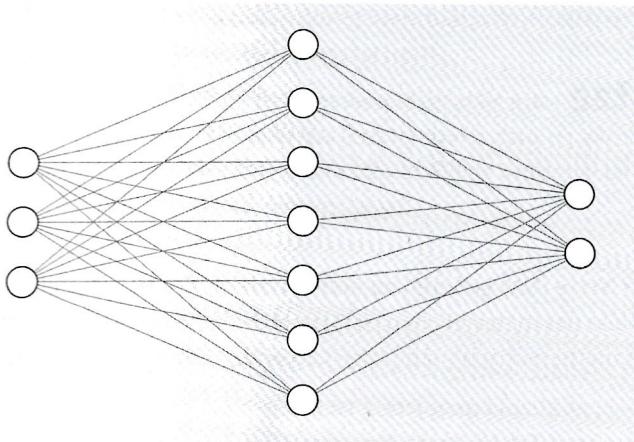


Abbildung 10.1 Ein typisches neuronales Netz, mit Eingabeschicht, versteckter Schicht und Ausgabeschicht

In diesem speziellen Beispiel besteht unsere Eingabe aus einer Reihe oder einem ~~Vek-~~tor mit drei Einträgen, also drei *numerischen* Größen. Die werden in der ~~versteckten~~ Schicht mit acht Knoten verändert, bevor sie zu einem Ausgabevektor mit zwei Einträgen weitergegeben werden. Somit haben wir es hier mit einem Beispiel zu tun, das ~~aus~~ einer dreidimensionalen Eingabe zwei Größen prädizieren will. Obwohl es numerische Ausgaben sind, kann man sie natürlich auch für Klassifikationsaufgaben verwenden.

Ich werde diese mathematischen Manipulationen gleich besprechen. Aber ich ~~möchte~~ an dieser Stelle noch anbringen, dass wir nicht auf eine versteckte Schicht ~~beschränkt~~ sind. Man kann im Prinzip beliebig viele Schichten mit beliebig vielen Knoten einsetzen, je nach Bedarf. Einzig die Eingabe- und Ausgabeschicht sind begrenzt durch die ~~geg-~~benen Daten und die Prädiktionsaufgabe. Die Struktur des Netzwerkes bezeichnet ~~man~~ auch als *(Netz-)Architektur*.

### 10.3 Universelles Approximations-Theorem

Das *universelle Approximations-Theorem* besagt, dass Sie, so lange Sie genug ~~Kno-~~nen zur Verfügung haben, jede kontinuierliche Funktion mit jedem Netzwerk mit nur einer versteckten Schicht zu einem beliebigen Genauigkeitsgrad approximieren können. Die Netzarchitektur für diese einfache Aufgabe sehen Sie in Abbildung 10.2, mit einer Eingabe  $x$  und einer Ausgabe  $y$ . Um die Approximation zu verbessern, fügen wir eine versteckten Schicht einfach mehr Knoten hinzu.

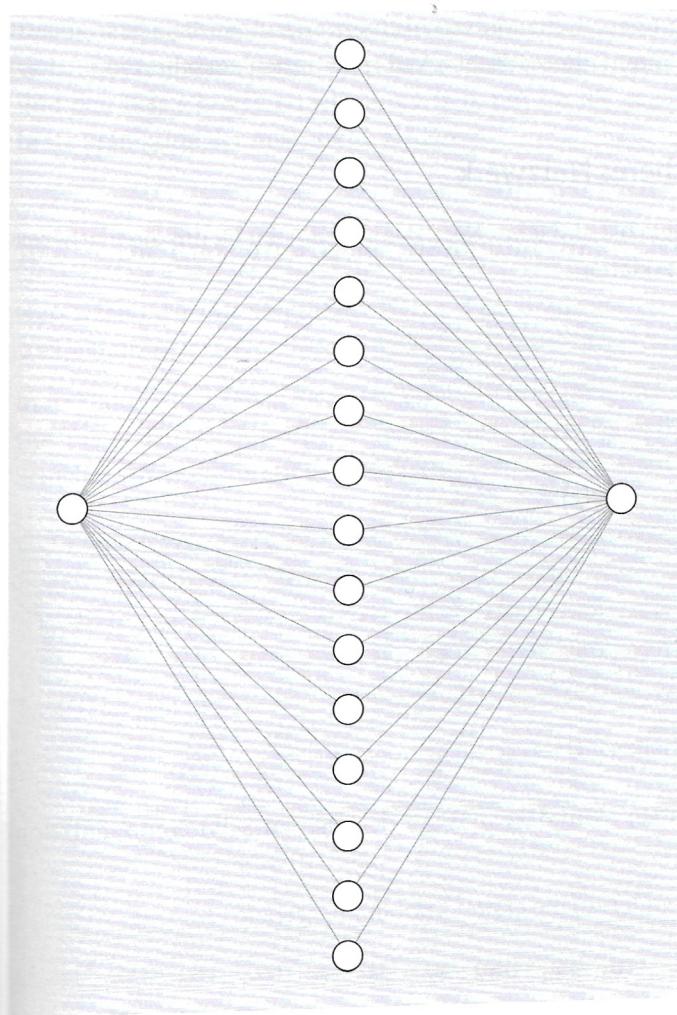


Abbildung 10.2 Neuronales Netz für das universelle Approximation-Theorem

Das ist einer der wichtigsten Einsatzbereiche neuronaler Netze, die Funktionsapproximation. Aber typischerweise sind unsere Probleme nicht so einfach formuliert wie hier implizit angenommen. Statt einer einzigen unabhängigen Variablen  $x$  haben wir eine ganze Reihe davon. Auch die Ausgabe  $y$  besteht aus mehreren Einträgen. Und wichtig, statt der Approximation einer bekannten Funktion müssen wir mit einer Riesenmenge an Daten, Eingaben und Ausgaben, zurechtkommen. Außerdem wollen wir die gefundene Approximation dann auf neue, noch ungesiehene Daten anwenden. Mit solch komplexen Herausforderungen braucht es dann doch eine umfangreichere Struktur mit mehr als einer versteckten Schicht.

Okay, aber jetzt wird es Zeit, uns anzuschauen, was in einer versteckten Schicht so ~~abla~~ und wie die von mir erwähnten mathematischen Manipulationen aussehen.

## 10.4 Ein noch einfacheres Netzwerk

Abbildung 10.3 zeigt ein möglichst einfaches Netz, das aber ausreicht, um die ~~matheme~~ tischen Manipulationen zu erklären.

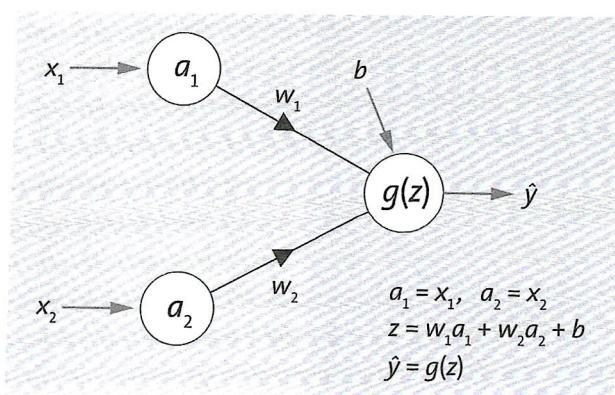


Abbildung 10.3 Die Manipulationen im klassischen neuronalen Netz

Links haben wir zwei Eingaben  $x_1$  und  $x_2$  und auf der rechten Seite eine Prädiktio~~n~~ Ausgabe  $\hat{y}$ .

Die Eingaben werden unverändert in die ersten Knoten übergeben:

$$a_1 = x_1 \quad \text{und} \quad a_2 = x_2$$

Dann wird jeder dieser Werte mit einem Gewicht multipliziert, dem jeweiligen  $w$ . Und ein Bias  $b$  wird dazuaddiert:

$$z = w_1 a_1 + w_2 a_2 + b$$

Das ist einfach nur eine Linearkombination.

Das Resultat wird dann einer Funktion  $g(z)$  übergeben und liefert uns die Ausgabe:

$$\hat{y} = g(z)$$

Wir haben hier eine sehr simple Transformation entwickelt und auf eine zweidimensionale Eingabe angewandt.

## Die mathematische Grundausbildung zum maschinellen Lernen

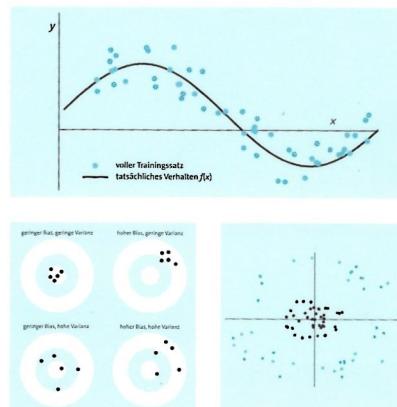
**Maschinelles Lernen** ist in aller Munde. Dieses Lehrbuch lenkt den Blick direkt auf den Kern der Sache – und das ist, aller Software-Frameworks zum Trotz, die Mathematik.

Jedes Lernverfahren wird Schritt für Schritt erklärt. Mit praktischen Tipps, vielen Zwischenschritten und einer Prise Humor. Die Mathematik ist für Studienanfänger nachvollziehbar.

Beim Trainieren der Modelle steckt der Teufel im Detail. Paul Wilmott zeigt, worauf es ankommt. Mit anschaulichen Beispielen von Natural Language Processing bis zum Abstimmungsverhalten im Parlament.

### Die Lernverfahren:

- + k-nächste Nachbarn
- + k-Means-Clustering
- + Naiver Bayes-Klassifikator
- + Lineare und logistische Regression
- + Support-Vektor-Maschinen
- + Selbstorganisierende Karten
- + Entscheidungsbäume
- + Neuronale Netze
- + Reinforcement Learning



**Paul Wilmott** vermittelt angewandte Mathematik – mit Kultstatus! Seine unverwechselbaren Einführungen bringen seit Jahrzehnten Licht in finanzmathematische Modelle und KI.

