Kapitel 1

Wie Computer aus Daten lernen können

Unserer Ansicht nach ist *das Machine Learning (maschinelles Lernen)*, die Anwendung und Wissenschaft von Algorithmen, die den Sinn von Daten erkennen können, das spannendste Forschungsfeld der Informatik! Wir leben in einem Zeitalter, in dem Daten im Überfluss vorhanden sind – und mit den selbstlernenden Algorithmen des Machine Learnings können wir diese Daten in Wissen verwandeln. Dank der vielen in den letzten Jahren entwickelten Open-Source-Bibliotheken ist jetzt der richtige Zeitpunkt gekommen, um sich eingehend mit dem Thema Machine Learning zu befassen und zu erfahren, wie leistungsfähige Algorithmen dafür eingesetzt werden können, Muster in den Daten zu erkennen und Vorhersagen über zukünftige Ereignisse zu treffen.

In diesem Kapitel werden wir die grundlegenden Konzepte und verschiedene Arten des Machine Learnings erörtern. Mit einer Einführung in die relevante Terminologie schaffen wir die Grundlage dafür, Verfahren des Machine Learnings erfolgreich zum Lösen von Aufgaben in der Praxis einzusetzen.

Dieses Kapitel hat folgende Themen zum Inhalt:

- Allgemeine Konzepte des Machine Learnings
- Die drei Arten des Machine Learnings und grundlegende Begriffe
- Die Bausteine des erfolgreichen Designs von Lernsystemen
- Installation von Python und Einrichtung einer für die Analyse von Daten und Machine Learning geeigneten Umgebung

1.1 Intelligente Maschinen, die Daten in Wissen verwandeln

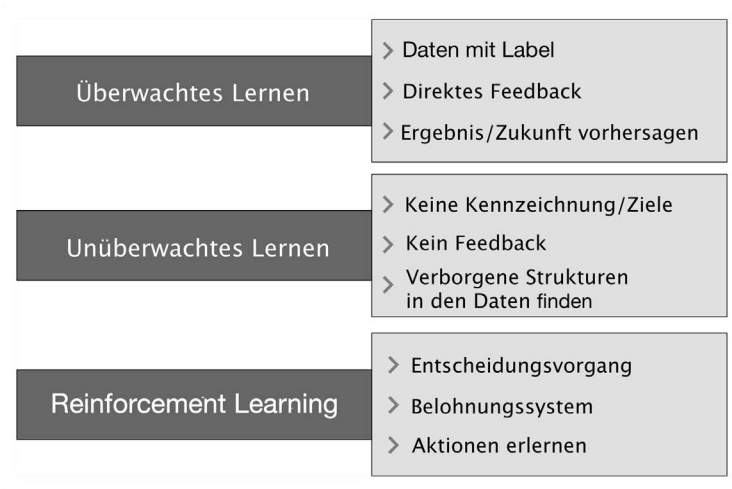
In diesem Zeitalter der modernen Technologie steht eine Ressource im Überfluss zur Verfügung: große Mengen von strukturierten und unstrukturierten Daten. In der zweiten Hälfte des 20. Jahrhunderts hat sich das Machine Learning als eine Teildisziplin der *Artificial Intelligence* (künstliche Intelligenz, KI) herausgebildet, bei der es um die Entwicklung selbstlernender Algorithmen geht, die Erkenntnisse aus Daten extrahieren, um bestimmte Vorhersagen treffen zu können. Das Erfordernis menschlichen Eingreifens zur manuellen Ableitung von Regeln und

der Entwicklung von Modellen anhand der Analyse großer Datenmengen erübrigt sich damit mehr und mehr, denn das Machine-Learning-Verfahren bietet eine effiziente Alternative zur Erfassung des in den Daten enthaltenen Wissens – die zudem die auf diesen Daten basierende Entscheidungsfindung sowie die Aussagekraft von Vorhersagemodellen zusehends verbessert.

Dieses Verfahren wird nicht nur in der Forschung immer wichtiger, es spielt auch im Alltag eine zunehmend größere Rolle: Dank des Machine Learnings erfreuen wir uns stabiler E-Mail-Spamfilter, praktischer Text- und Spracherkennungssoftware, verlässlicher Suchmaschinen, kaum zu schlagender Schachcomputer und hoffentlich bald auch sicherer selbstfahrender Autos. Auch bei medizinischen Anwendungen hat es bemerkenswerte Fortschritte gegeben. So haben Forscher beispielsweise demonstriert, dass Deep-Learning-Modelle Hautkrebs fast so gut wie Menschen erkennen können (`https://www.nature.com/articles/nature21056`). Bei DeepMind haben Forscher kürzlich einen weiteren Meilenstein erreicht. Sie konnten mithilfe von Deep Learning die dreidimensionale Struktur von Proteinen vorhersagen und haben dabei erstmals bessere Ergebnisse als mit physikalischen Ansätzen erzielt (`https://deepmind.com/blog/alphafold/`).

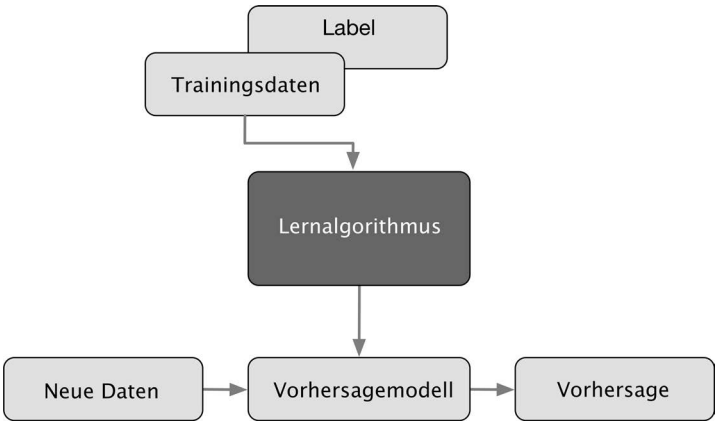
1.2 Die drei Arten des Machine Learnings

In diesem Abschnitt werden wir die drei verschiedenen Gattungen des Machine Learnings betrachten: *überwachtes Lernen*, *unüberwachtes Lernen* und *Reinforcement Learning*. Sie werden erfahren, welche grundlegenden Unterschiede es zwischen diesen drei Varianten gibt und anhand von Beispielen allmählich ein Gespür dafür entwickeln, auf welche praktischen Aufgabenstellungen sie sich anwenden lassen:



1.2.1 Mit überwachtem Lernen Vorhersagen treffen

Das Hauptziel des überwachten Lernens ist, ein Modell anhand mit Labels gekennzeichneter *Trainingsdaten* zu erlernen, um so Voraussagen über unbekannte oder zukünftige Daten treffen zu können. Der Begriff *»überwacht«* bezieht sich hier auf Trainingsdaten (Eingabedaten), die bereits mit den bekannten erwünschten Ausgabewerten (Bezeichnungen/Labels) gekennzeichnet sind.



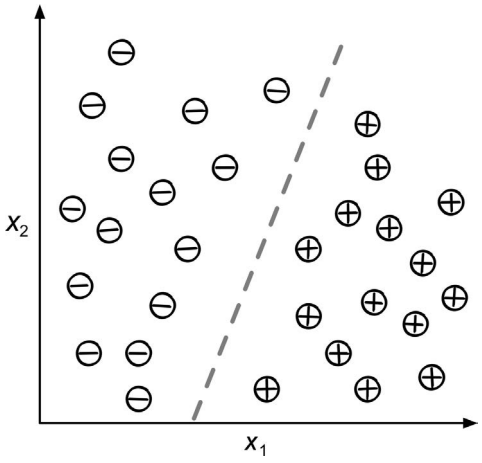
Betrachten wir als Beispiel das Filtern von E-Mail-Spam. Wir können einen überwachten Lernalgorithmus mit einer Sammlung von als Spam oder Nicht-Spam gekennzeichneten E-Mails »trainieren«, um dann vorherzusagen, zu welcher dieser Klassen eine neue E-Mail gehört. Eine solche Einteilung in bestimmte Klassen wird als *Klassifikation* bezeichnet. Eine weitere Unterkategorie des überwachten Lernens ist die *Regression*, bei der die Ausgabewerte im Gegensatz zur Klassifikation stetig sind.

Klassifikation: Vorhersage der Klassenbezeichnungen

Die Klassifikation ist eine Unterkategorie des überwachten Lernens, die es zum Ziel hat, anhand vorhergehender Beobachtungen die kategorialen Klassen neuer Instanzen vorherzusagen. Die Bezeichnungen dieser Klassen sind eindeutige, ungeordnete Werte, die als Gruppenzugehörigkeit der Instanzen aufgefasst werden können. Die soeben erwähnte E-Mail-Spamerkennung stellt ein typisches Beispiel für eine *binäre Klassifikation* dar, denn der Algorithmus erlernt Regeln, um zwischen zwei möglichen Klassen zu unterscheiden: Spam oder Nicht-Spam.

Die folgende Abbildung illustriert das Konzept einer binären Klassifikation, die mit 30 Beispielen trainiert wird, von denen 15 als *negative Klasse* (Minuszeichen) und weitere 15 als *positive Klasse* (Pluszeichen) gekennzeichnet sind. Die Datenmenge ist in diesem Szenario zweidimensional: Jedem Beispiel sind die beiden

Werte x_1 und x_2 zugeordnet. Nun können wir dem überwachten Lernalgorithmus eine Regel beibringen: Die durch eine gestrichelte Linie dargestellte Grenze trennt die beiden Klassen voneinander und ermöglicht es, neue Daten anhand der Werte von x_1 und x_2 einer der beiden Klassen zuzuordnen.



Die Anzahl der Klassenbezeichnungen muss allerdings nicht auf zwei beschränkt sein. Das von einem überwachten Lernalgorithmus erlernte Vorhersagemodell kann einer neuen, noch nicht mit Label gekennzeichneten Instanz jede Bezeichnung zuordnen, die in den Trainingsdaten vorkommt.

Ein typisches Beispiel für solch eine *Multiklassen-Klassifikation* ist die Handschrifterkennung. Hier könnten wir eine Trainingsdatenmenge zusammenstellen, die aus mehreren handgeschriebenen Beispielen aller Buchstaben des Alphabets besteht. Die Buchstaben (»A«, »B«, »C« usw.) repräsentieren die verschiedenen Kategorien oder Klassenbezeichnungen, die wir vorhersagen möchten. Wenn dann ein Anwender über ein Eingabegerät einen neuen Buchstaben angibt, wäre unser Vorhersagemodell in der Lage, diesen mit einer gewissen Zuverlässigkeit zu erkennen. Das System wäre allerdings nicht imstande, irgendeine der Ziffern von null bis neun zu erkennen, sofern diese nicht ebenfalls Bestandteil der Trainingsdaten waren.

Regression: Vorhersage stetiger Ergebnisse

Im vorangegangenen Abschnitt haben wir festgestellt, dass es die Aufgabe einer Klassifikation ist, Instanzen kategoriale, ungeordnete Klassenbezeichnungen zuzuordnen. Ein zweiter Typ des überwachten Lernens ist die Vorhersage stetiger Ergebnisse, die auch als *Regressionsanalyse* bezeichnet wird. Hierbei sind verschiedene unabhängige oder *erklärende* Variablen sowie eine stetige Zielvariable (Ergebnis) vorgegeben und wir versuchen, eine Beziehung zwischen diesen Variablen zu finden, um Ergebnisse vorhersagen zu können.

Beachten Sie hier, dass die erklärenden Variablen beim Machine Learning oft als »Merkmale« oder »Features« und die Ergebnisse als »Zielvariablen« bezeichnet werden. Wir werden diese Begriffe ebenfalls verwenden.

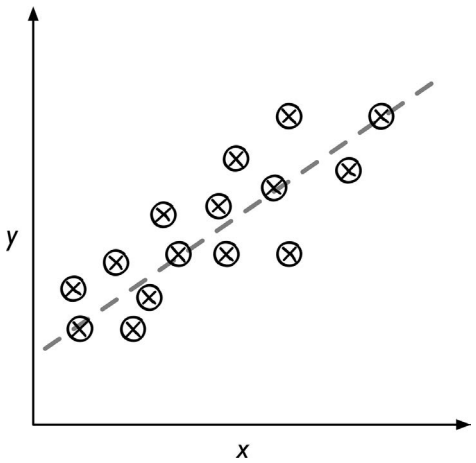
Nehmen wir beispielsweise an, dass wir die von Schülern bei einer Matheprüfung erreichten Punktzahlen prognostizieren möchten. Sofern es einen Zusammenhang zwischen der mit dem Üben für die Prüfung verbrachten Zeit und den erzielten Punktzahlen gibt, könnten wir daraus Trainingsdaten für ein Modell herleiten, das anhand der aufgewendeten Übungszeit die Punktzahlen von Schülern voraussagt, die die Prüfung in Zukunft ebenfalls abzulegen beabsichtigen.

Tipp: Regression zur Mitte

Der Begriff *Regression* wurde schon 1886 von Francis Galton in einem Artikel mit dem Titel *Regression Towards Mediocrity in Hereditary Stature* geprägt. Galton beschrieb darin das Phänomen, dass sich bei der Bevölkerung die mittlere Abweichung von der durchschnittlichen Körpergröße im Laufe der Zeit nicht vergrößert.

Er beobachtete, dass die Körpergröße der Eltern nicht an die Kinder vererbt wird, vielmehr nähert sich die Größe der Kinder dem Durchschnittswert an.

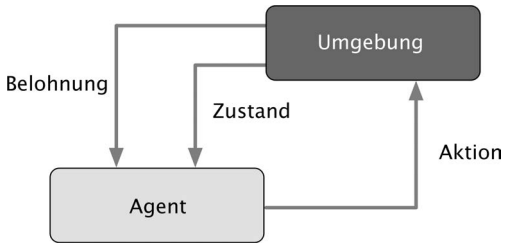
Die folgende Abbildung illustriert das Konzept der *linearen Regression*. Bei vorgegebener unabhängiger Variablen x und abhängiger Variablen y passen wir eine Gerade so an die Daten an, dass ein Maß für den Abstand der Geraden von den Beispielwerten (üblicherweise der Mittelwert der quadrierten Differenzen) minimal wird. Nun können wir den aus den Daten ermittelten Schnittpunkt mit der y-Achse sowie die Steigung der Geraden verwenden, um das Ergebnis für neue Werte vorherzusagen.



1.2.2 Interaktive Aufgaben durch Reinforcement Learning lösen

Die dritte Variante des Machine Learnings ist das Reinforcement Learning. Hier besteht die Zielsetzung darin, ein System zu entwickeln (den *Agenten*), das seine Leistung durch Interaktionen mit seiner *Umgebung* verbessert. Zu den Informationen über den aktuellen Zustand der Umgebung gehört typischerweise ein sogenanntes *Belohnungssignal*, daher ist das Reinforcement Learning in gewisser Weise mit dem überwachten Lernen verwandt. Allerdings handelt es sich bei diesem Feedback nicht um die korrekte Klassenbezeichnung oder den richtigen Wert, sondern um eine Bewertung dafür, wie gut die Aktion war, dies wird durch eine *Belohnungsfunktion* festgelegt. Der Agent kann so über Interaktionen mit seiner Umgebung durch Reinforcement Learning erkennen, welche Aktionen besonders gut belohnt werden. Das kann durch schlichtes Ausprobieren (Versuch und Irrtum) oder durch bewusste Planung geschehen.

Ein schönes Beispiel für Reinforcement Learning ist ein Schachcomputer. Hier bewertet der Agent nach einer Reihe von Zügen die Stellung auf dem Schachbrett (die Umgebung), und die Belohnung kann am Ende des Spiels als *Sieg* oder *Niederlage* definiert werden.



Es gibt eine Vielzahl verschiedener Unterarten des Reinforcement Learnings. Im Allgemeinen versucht der Agent jedoch, die Belohnung durch eine Reihe von Interaktionen mit der Umgebung zu maximieren. Jedem Zustand kann eine positive (oder negative) Belohnung zugeordnet werden, und diese Belohnung kann dadurch definiert werden, dass ein Gesamtziel erreicht wird, wie z.B. das Gewinnen oder das Verlieren einer Schachpartie. Beim Schachspiel kann etwa das Ergebnis eines jeden Spielzugs als ein anderer Zustand der Umgebung aufgefasst werden.

Um beim Schach zu bleiben: Stellen Sie sich das Erreichen bestimmter Stellungen auf dem Schachbrett als positives Ereignis vor, das es wahrscheinlicher macht, das Spiel zu gewinnen – beispielsweise das Schlagen einer gegnerischen Spielfigur oder das Bedrohen der Dame. Andere Stellungen wiederum werden als negativ erachtet, beispielsweise wenn eine der eigenen Spielfiguren beim nächsten Zug geschlagen werden kann. Nun erfolgt die Belohnung (positive beim Gewinnen und negative beim Verlieren) beim Schach natürlich erst am Ende des Spiels. Darüber hinaus hängt die Belohnung auch davon ab, wie der Gegner spielt. Er könnte beispielsweise die Dame opfern, das Spiel aber trotzdem gewinnen. Das Reinforce-

ment Learning versucht, eine Reihe von Aktionen zu erlernen, die die Belohnung insgesamt maximieren – entweder durch eine sofortige Belohnung nach einem Zug, oder aber durch eine *verzögerte* Belohnung.

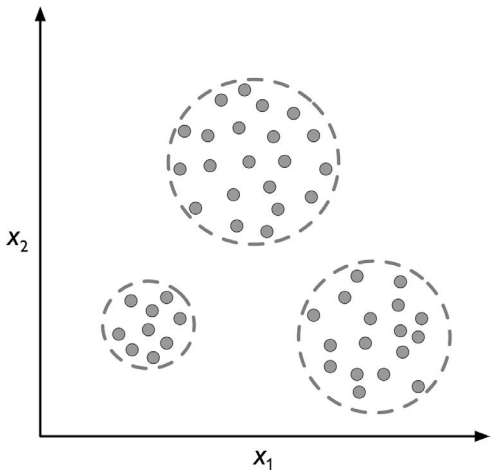
1.2.3 Durch unüberwachtes Lernen verborgene Strukturen erkennen

Beim überwachten Lernen ist die richtige Antwort beim Trainieren des Modells bereits im Vorhinein bekannt, und beim Reinforcement Learning definieren wir eine Bewertung oder *Belohnung* für bestimmte Aktionen des Agenten. Beim unüberwachten Lernen hingegen haben wir es mit nicht mit Label gekennzeichneten Daten oder mit Daten unbekannter Struktur zu tun. Durch die beim unüberwachten Lernen eingesetzten Verfahren sind wir in der Lage, die Struktur der Daten zu erkunden, um sinnvolle Informationen daraus zu extrahieren, ohne dass es Hinweise auf eine Zielvariable oder eine Belohnungsfunktion gibt.

Bestimmung von Untergruppen durch Clustering

Clustering ist ein exploratives Datenanalyseverfahren, das es uns gestattet, Informationen in sinnvolle Untergruppen (*Cluster*) aufzuteilen, ohne vorherige Kenntnisse über die Gruppenzugehörigkeit dieser Informationen zu besitzen. Jeder bei der Analyse auftretende Cluster definiert eine Gruppe von Objekten, die bestimmte Eigenschaften gemeinsam haben, sich aber von Objekten in anderen Gruppen hinreichend unterscheiden. Deshalb wird das Clustering manchmal auch als *unüberwachte Klassifikation* bezeichnet. Es ist ausgezeichnet geeignet, um Informationen zu strukturieren und sinnvolle Beziehungen zwischen den Daten abzuleiten. Beispielsweise ermöglicht es Marketingfachleuten, Kunden anhand ihrer Interessen in Gruppen einzuordnen, um gezielte Kampagnen zu entwickeln.

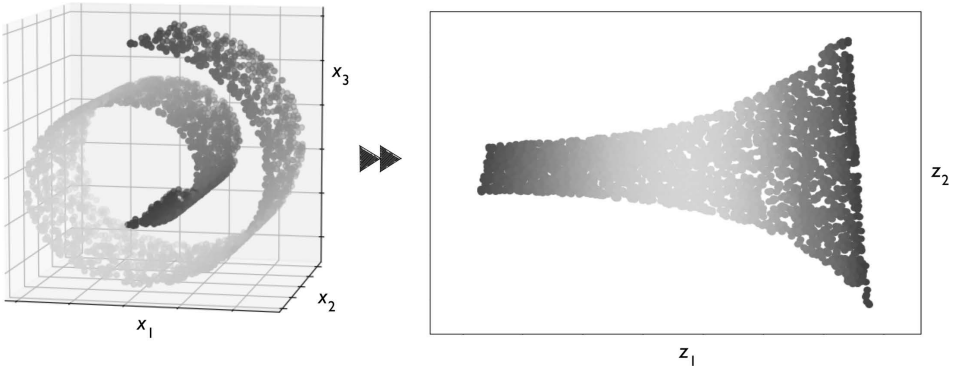
Die folgende Abbildung illustriert, wie man das Clustering-Verfahren zur Organisation nicht mit Label gekennzeichneter Daten in drei verschiedene Gruppen verwenden kann, die jeweils ähnliche Merkmale x_1 und x_2 aufweisen.



Datenkomprimierung durch Dimensionsreduktion

Die *Dimensionsreduktion* ist eine weitere Teildisziplin des unüberwachten Lernens. Wir haben es des Öfteren mit Daten hoher Dimensionalität zu tun (jede Beobachtung besteht aus einer Vielzahl von Messwerten), was aufgrund der für die Lernalgorithmen geltenden Beschränkungen von Speicherplatz und Rechenleistung eine Herausforderung darstellen kann. Bei der Vorverarbeitung von Merkmalen wird häufig eine unüberwachte Dimensionsreduktion eingesetzt, um die Daten von sogenanntem »Rauschen« zu befreien. Dies kann allerdings zu einer Abschwächung der Aussagekraft bestimmter Vorhersagealgorithmen führen. Die Daten werden in kleinere Unterräume geringerer Dimensionalität aufgeteilt, wobei der Großteil der relevanten Informationen erhalten bleibt.

In manchen Fällen ist die Dimensionsreduktion auch für die Visualisierung der Daten nützlich. Beispielsweise können hochdimensionale Merkmalsmengen auf ein-, zwei- oder dreidimensionale Merkmalsräume projiziert werden, um sie als 3-D- oder 2-D-Streudiagramme bzw. -Histogramme darzustellen. Die Abbildung zeigt ein Beispiel, in dem eine nichtlineare Dimensionsreduktion auf eine 3-D-Punktmenge in Form einer Biskuitrolle angewendet wurde, um sie in einen zweidimensionalen Merkmalsraum zu transformieren.



1.3 Grundlegende Terminologie und Notation

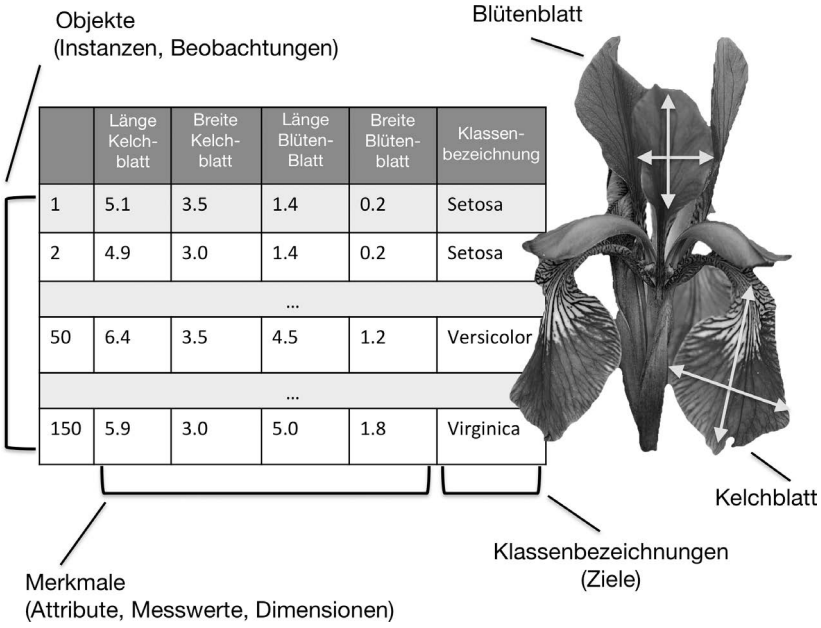
Nachdem wir nun die drei Arten des Machine Learnings – überwachtes und unüberwachtes Lernen sowie Reinforcement Learning – erörtert haben, werden wir als Nächstes die grundlegenden Begriffe klären, die in den folgenden Kapiteln Verwendung finden. Der folgende Abschnitt erläutert die Begriffe, die für die Beschreibung der verschiedenen Aspekte einer Datenmenge verwendet werden, sowie die mathematische Notation, die zur präzisen Beschreibung zum Einsatz kommt.

Da Machine Learning ein so umfassendes und interdisziplinäres Fachgebiet ist, werden Sie früher oder später mit Sicherheit vielen verschiedenen Begriffen begegnen, die ein und dasselbe Konzept beschreiben. Im zweiten der nachfolgenden Abschnitte sind die am häufigsten verwendeten Begriffe aufgeführt, die sich in der

Literatur zum Thema Machine Learning finden. Sie erweisen sich bei der weiteren Lektüre vielleicht als nützlich.

1.3.1 Im Buch verwendete Notation und Konventionen

Die folgende Abbildung zeigt einen Auszug der *Iris-Datensammlung*, einem klassischen Beispiel für den Bereich des Machine Learnings. Dabei handelt es sich um Messdaten von 150 Schwertlilien dreier verschiedener Arten: *Iris setosa*, *Iris versicolor* und *Iris virginica*. Jedes der Blumenexemplare wird in dieser Datensammlung durch eine Zeile repräsentiert. In den einzelnen Spalten stehen die in Zentimetern angegebenen Messdaten, die wir auch als *Merkmale* der Datenmenge bezeichnen.



	Länge Kelch-blatt	Breite Kelch-blatt	Länge Blüten-Blatt	Breite Blüten-blatt	Klassen-bezeichnung
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Um die Notation und Implementierung einfach, aber dennoch effizient zu halten, nutzen wir die Grundlagen der *linearen Algebra*. In den nachfolgenden Kapiteln verwenden wir die Matrizen- und Vektornotation zur Beschreibung der Daten. Wir folgen der üblichen Konvention, dass jedes Objekt durch eine Zeile in der Merkmalsmatrix $\boldsymbol{X}$ repräsentiert und jedes Merkmal als eigene Spalte gespeichert wird.

Die Iris-Datensammlung besteht aus 150 Datensätzen mit jeweils vier Merkmalen und kann somit als 150×4-Matrix $\boldsymbol{X} \in \mathbb{R}^{150\times4}$ geschrieben werden:

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{bmatrix}$$

Ab jetzt verwenden wir ein hochgestelltes `i` und ein tiefgestelltes `j`, um auf das *i*-te Trainingsobjekt bzw. die *j*-te Dimension der Trainingsdatenmenge zu verweisen.

Wir notieren Vektoren $\left(\boldsymbol{x} \in \mathbb{R}^{n\times 1}\right)$ als fett gedruckte Kleinbuchstaben und Matrizen ($\boldsymbol{X} \in \mathbb{R}^{n\times m}$) als fett gedruckte Großbuchstaben. Um auf einzelne Elemente eines Vektors oder einer Matrix zu verweisen, werden kursive Buchstaben benutzt ($x^{(n)}$ bzw. $x^{(n)}_{(m)}$).

Beispielsweise verweist x_1^{150} auf die erste Dimension des Blumenexemplars 150, die Länge des Kelchblatts. Jede Zeile der Merkmalsmatrix repräsentiert ein Blumenexemplar und kann als vierdimensionaler Zeilenvektor $\boldsymbol{x}^{(i)} \in \mathbb{R}^{1\times 4}$ geschrieben werden, z.B.:

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} & x_4^{(i)} \end{bmatrix}$$

Jede Merkmalsdimension ist ein 150-dimensionaler Spaltenvektor $\boldsymbol{x}_j \in \mathbb{R}^{150\times 1}$:

$$\boldsymbol{x}_j = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \vdots \\ x_j^{(150)} \end{bmatrix}$$

Die Zielvariablen (hier die Klassenbezeichnungen) werden ebenfalls als 150-dimensionale Spaltenvektoren notiert:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(150)} \end{bmatrix} \left(y \in \{\text{Setosa},\ \text{Versicolor},\ \text{Virginica}\}\right)$$

1.3.2 Terminologie

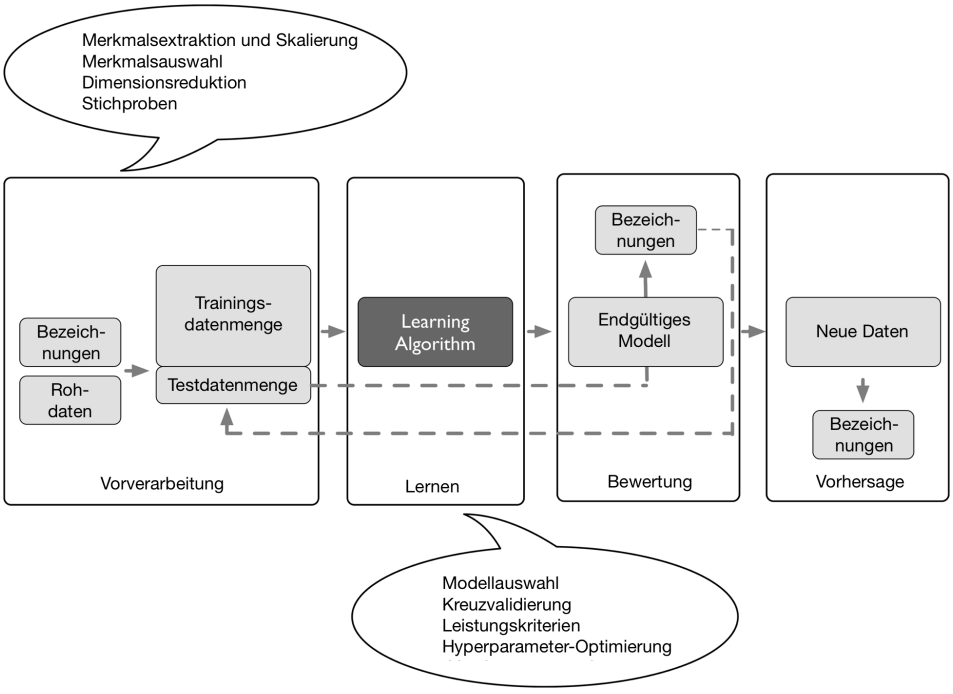
Wie erwähnt ist Machine Learning ein weites Feld und ausgeprägt interdisziplinär, denn es bringt Forscher aus den verschiedensten Fachgebieten zusammen. Viele Konzepte und Begriffe wurden neu entdeckt oder umdefiniert oder sind Ihnen vielleicht schon unter einem anderen Namen bekannt. In der folgenden Liste finden Sie eine Auswahl gebräuchlicher Begriffe und ihrer Synonyme, die sich bei der Lektüre dieses Buches oder anderer Bücher über Machine Learning vielleicht als nützlich erweisen.

- Trainingsbeispiel: Eine Zeile in einer Tabelle, die eine Datenmenge repräsentiert. Der Begriff ist gleichbedeutend mit Beobachtung, Datensatz, Instanz oder Stichprobe (mit einer Stichprobe sind für gewöhnlich mehrere Trainingsbeispiele gemeint).

- Training: Anpassung des Modells; bei parametrischen Modellen entspricht das einer Parameterschätzung.
- Merkmal, kurz *x*: Eine Spalte in einer Datentabelle oder einer Datenmatrix. Synonyme: Feature, unabhängige Variable, Eingabe, Attribut oder Kovariate.
- Ziel, kurz *y*: Synonyme: Ergebnis, Ausgabe, Antwort-/Ausgabevariable, abhängige Variable oder (Klassen-)Label oder (Klassen-)Bezeichnung
- Verlustfunktion: Kurz Verlust, wird manchmal als Synonym für Kostenfunktion bzw. Straffunktion gebraucht. Mitunter wird sie auch als Error- oder Fehlerfunktion bezeichnet (nicht zu verwechseln mit der gaußschen Fehlerfunktion). Der Begriff »Verlust« bezieht sich auf den Verlust, der für einen einzelnen Datenpunkt ermittelt wird. Die Kosten hingegen sind ein Maß für den Verlust der gesamten Datenmenge (gemittelt oder summiert).

1.4 Entwicklung eines Systems für das Machine Learning

In den vorangegangenen Abschnitten haben wir die grundlegenden Konzepte des Machine Learnings und die drei verschiedenen Arten des Lernens erörtert. In diesem Abschnitt befassen wir uns mit weiteren wichtigen Bestandteilen eines Systems für dieses Verfahren, die den Lernalgorithmus begleiten. Das folgende Diagramm zeigt den typischen Ablauf, der beim Machine Learning in Vorhersagemodellen zum Einsatz kommt, die wir in den folgenden Abschnitten betrachten werden.



1.4.1 Vorverarbeitung: Daten in Form bringen

Rohdaten liegen nur selten in einer für die optimale Leistung eines Lernalgorithmus erforderlichen Form vor, deshalb ist die *Vorverarbeitung* der Daten bei jedem Lernalgorithmus von entscheidender Bedeutung.

Im Fall der Iris-Datensammlung aus dem vorangegangenen Abschnitt könnten die Rohdaten beispielsweise als eine Reihe von Fotos der Blumenexemplare vorliegen, denen wir sinnvolle Merkmale entnehmen möchten. Das könnten etwa Grundfarbe und Tönung sowie Höhe, Länge und Breite der Pflanzen sein.

Bei vielen Lernalgorithmen ist es außerdem erforderlich, dass die ausgewählten Merkmale irgendwie normiert sind (hier müssten die Pflanzen im selben Maßstab dargestellt sein), um ein optimales Ergebnis zu erzielen. Dies wird oftmals dadurch erreicht, dass die ausgewählten Merkmale auf ein Intervall `[0,1]` oder eine Standardnormalverteilung (Mittelwert 0 und Standardabweichung 1) abgebildet werden, wie Sie in den nachfolgenden Kapiteln noch sehen werden.

Manche der ausgewählten Merkmale könnten hochgradig korreliert und daher in gewissem Maße redundant sein. In diesen Fällen sind Verfahren zur Dimensionsreduktion nützlich, um die Merkmale auf einen Merkmalsraum geringer Dimensionalität abzubilden. Die Dimensionsreduktion des Merkmalraums hat die Vorteile, dass weniger Speicherplatz benötigt wird und der Lernalgorithmus erheblich schneller arbeitet. In manchen Fällen kann eine Dimensionsreduktion auch die Vorhersagekraft eines Modells verbessern, nämlich wenn die Datenmenge eine große Anzahl irrelevanter Merkmale (Rauschen) aufweist, das heißt, dass sie ein niedriges Signal-zu-Rausch-Verhältnis besitzt.

Um festzustellen, ob ein Lernalgorithmus nicht nur die Trainingsdaten ordentlich verarbeitet, sondern auch mit neuen Daten gut zurechtkommt, ist es sinnvoll, den Datenbestand nach dem Zufallsprinzip in separate Trainings- und Testdatenmengen aufzuteilen: Zum Trainieren und Optimieren des Lernmodells verwenden wir die Trainingsdatenmenge, während wir die Testdatenmenge bis zum Schluss zurückhalten, um das endgültige Modell bewerten zu können.

1.4.2 Trainieren und Auswählen eines Vorhersagemodells

Wie Sie in den nachfolgenden Kapiteln noch sehen werden, sind viele verschiedene Lernalgorithmen entwickelt worden, mit denen die unterschiedlichsten Aufgabenstellungen erledigt werden können. An dieser Stelle ist es allerdings wichtig festzuhalten, dass das Lernen nicht umsonst zu haben ist – so in etwa könnte man David Wolperts berühmte »No Free Lunch«-Theoreme zusammenfassen (*The Lack of A Priori Distinctions Between Learning Algorithms*, D.H. Wolpert 1996; *No Free Lunch Theorems for Optimization*, D.H. Wolpert und W.G. Macready, 1997). Noch besser lässt sich dieses Konzept anhand eines berühmten Zitats veranschaulichen:

»Wenn das einzige verfügbare Werkzeug ein Hammer ist, dürfte es verlockend sein, alles wie einen Nagel zu behandeln.« (Abraham Maslow, 1966). Beispielsweise sind alle Klassifikationsalgorithmen in irgendeiner Weise voreingenommen und kein Klassifikationsmodell ist anderen überlegen, wenn man nicht bestimmte Annahmen über die Aufgabenstellung macht. In der Praxis ist es daher von entscheidender Bedeutung, wenigstens eine Handvoll verschiedener Algorithmen zu vergleichen, um das am besten funktionierende Modell zu trainieren und auszuwählen. Aber um Vergleiche zwischen verschiedenen Modellen anstellen zu können, müssen zunächst einmal Bewertungskriterien festgelegt werden. Ein gebräuchliches Kriterium ist die *Korrektklassifikationsrate* (kurz: Klassifikationsrate, engl. *accuracy*, ACC) des Modells, die als Anteil der korrekten Klassifikationen definiert ist.

Nun stellt sich natürlich die Frage: *Wie kann man wissen, welches Modell mit den Testdaten und den »echten« Daten gut funktioniert, wenn man sie nicht bei der Auswahl des Modells verwendet, sondern bis zur Bewertung des endgültigen Modells zurückhält?* Um das mit dieser Frage verbundene Problem zu lösen, können verschiedene Kreuzvalidierungsverfahren eingesetzt werden, bei denen die Trainingsdatenmenge weiter in Trainings- und *Validierungsteilmengen* aufgeteilt wird, um die *Generalisierungsfähigkeit* des Modells abzuschätzen. Und schließlich dürfen wir auch nicht erwarten, dass die Standardparameter der Lernalgorithmen verschiedener Softwarebibliotheken für unsere spezielle Aufgabenstellung optimiert sind. Daher werden wir in den noch folgenden Kapiteln häufig Gebrauch von Verfahren zur *Hyperparameter-Optimierung* machen, um die Leistung unseres Modells feiner abzustimmen.

Man kann sich diese Hyperparameter als Parameter vorstellen, die nicht anhand der Daten ermittelt werden, sondern als Einstellungsmöglichkeiten zur Verbesserung der Leistung, was sehr viel klarer werden wird, wenn wir uns in den folgenden Kapiteln einige dazu passende Beispiele ansehen.

1.4.3 Bewertung von Modellen und Vorhersage anhand unbekannter Dateninstanzen

Nach der Auswahl eines an die Trainingsdaten angepassten Modells können wir die Testdatenmenge verwenden, um zu ermitteln, wie gut es mit diesen unbekannten Daten bei der Schätzung des Generalisierungsfehlers zurechtkommt. Sofern die Leistung des Modells zufriedenstellend ausfällt, können wir es verwenden, um anhand neuer, zukünftiger Daten Vorhersagen zu treffen. Hier muss angemerkt werden, dass die Parameter der vorhin erwähnten Verfahren (wie die Skalierung der Merkmalsdarstellungen oder die Dimensionsreduktion) ausschließlich anhand der Trainingsdatenmenge ermittelt werden. Dieselben Parameter werden später auch auf die Testdatenmenge und neue Daten angewendet – ansonsten könnte die bei den Testdaten gemessene Leistung zu optimistisch sein.

1.5 Machine Learning mit Python

Im Bereich Data Science ist Python eine der beliebtesten Programmiersprachen, daher gibt es eine Vielzahl nützlicher Bibliotheken, die von der sehr aktiven Python-Community entwickelt wurden.

Die Performance von Interpretersprachen wie Python ist derjenigen von kompilierten Programmiersprachen zwar unterlegen, es gibt allerdings Erweiterungsbibliotheken wie NumPy und SciPy, die auf maschinennahen Fortran- und C-Implementierungen beruhen, um schnelle vektorisierte Berechnungen mit mehrdimensionalen Arrays auszuführen.

Bei Aufgabenstellungen des Machine Learnings werden wir zumeist auf *scikit-learn* zurückgreifen, eine weitverbreitete und leicht verständliche Open-Source-Bibliothek für Machine Learning. In den nachfolgenden Kapiteln, in denen wir uns auf ein Teilgebiet des Machine Learnings namens *Deep Learning* konzentrieren, werden wir die neueste Version der TensorFlow-Bibliothek verwenden, die darauf spezialisiert ist, sogenannte tiefe neuronale Netze zu trainieren, indem sie die Rechenleistung von Grafikprozessoren nutzt.

1.5.1 Python und Python-Pakete installieren

Python ist für die drei wichtigsten Betriebssysteme Microsoft Windows, macOS und Linux verfügbar. Das Installationsprogramm und die Dokumentation stehen auf der offiziellen Website unter `https://www.python.org` zum Herunterladen bereit.

Dieses Buch setzt mindestens die Python-Version 3.7.0 voraus, es empfiehlt sich jedoch, immer die neueste verfügbare Python-3-Version zu verwenden. Einige der Codebeispiele sind möglicherweise auch mit Python-Versionen ab 2.7.0 kompatibel, aber da Python 2.7 seit 2019 offiziell nicht mehr unterstützt wird, was auch für die meisten Open-Source-Bibliotheken zutrifft (`https://python3statement.org`), raten wir dringend dazu, Python 3.7 oder neuer zu verwenden.

Die zusätzlichen Pakete, die wir im Buch benutzen werden, können mit *pip* installiert werden. Dieses Installationsprogramm gehört seit der Python-Version 3.3 zur Standardbibliothek. Weitere Informationen über `pip` finden Sie unter `https://docs.python.org/3/installing/index.html`.

Nach erfolgreicher Python-Installation können Sie mit `pip` wie folgt weitere Python-Pakete installieren:

```
pip install Paketname
```

Bereits installierte Pakete können mit der Option `--upgrade` aktualisiert werden:

```
pip install Paketname --upgrade
```

1.5.2 Verwendung der Python-Distribution Anaconda

Von Continuum Analytics gibt es eine sehr empfehlenswerte alternative Python-Distribution für wissenschaftliches Rechnen namens *Anaconda*. Hierbei handelt es sich um eine – auch für den kommerziellen Gebrauch – kostenlose Python-Distribution, die alle wichtigen Python-Pakete für Data Science, Mathematik und Engineering in einem einzigen, benutzerfreundlichen und plattformunabhängigen Paket bündelt. Das Installationsprogramm können Sie unter `https://docs.anaconda.com/anaconda/install/` herunterladen. Eine Kurzanleitung ist unter `https://docs.anaconda.com/anaconda/user-guide/getting-started/` verfügbar.

Nach der Installation von Anaconda können Python-Pakete mit dem folgenden Befehl installiert werden:

```
conda install Paketname
```

Bereits vorhandene Pakete werden so aktualisiert:

```
conda update Paketname
```

1.5.3 Pakete für wissenschaftliches Rechnen, Data Science und Machine Learning

Im weiteren Verlauf des Buches werden wir vornehmlich mehrdimensionale Arrays von NumPy verwenden, um Daten zu speichern und zu verarbeiten. Gelegentlich kommt auch *pandas* zum Einsatz, eine auf NumPy beruhende Bibliothek, die erweiterte Funktionen für die noch komfortablere Verarbeitung von Tabellendaten bereitstellt. Zur Ergänzung des Lernerlebnisses werden wir darüber hinaus die sehr anpassungsfähige Matplotlib-Bibliothek einsetzen, die für die Visualisierung und das intuitive Verständnis quantitativer Daten oft äußerst nützlich ist.

Die Versionsnummern der im Buch verwendeten Python-Pakete sind nachstehend aufgeführt. Vergewissern Sie sich, dass Ihre installierten Pakete mindestens diesen Versionsnummern entsprechen, damit gewährleistet ist, dass die Codebeispiele korrekt ausgeführt werden.

- NumPy 1.17.4
- SciPy 1.3.1
- scikit-learn 0.22.0
- matplotlib 3.1.0
- pandas 0.25.3

1.6 Zusammenfassung

In diesem Kapitel haben wir einen ganz allgemeinen Blick auf das Thema Machine Learning geworfen und uns mit dem Gesamtbild sowie den grundlegenden Konzepten vertraut gemacht, die wir in den folgenden Kapiteln eingehender betrachten werden. Sie haben erfahren, dass überwachtes Lernen aus zwei wichtigen Teilgebieten besteht: Klassifikation und Regression. Klassifikationsmodelle ermöglichen es, Objekte bekannten Klassen zuzuordnen und wir können die Regressionsanalyse nutzen, um stetige Werte einer Zielvariablen vorherzusagen. Das unüberwachte Lernen bietet nicht nur praktische Verfahren zum Auffinden von Strukturen in nicht mit Label gekennzeichneten Daten, es kann darüber hinaus bei der Vorverarbeitung auch zur Datenkomprimierung eingesetzt werden.

Wir haben uns kurz die typische Vorgehensweise bei der Anwendung des Machine Learnings auf Problemstellungen angesehen, die bei der weiteren Erörterung und für praktische Beispiele in den folgenden Kapiteln als Grundlage dient. Darüber hinaus haben wir unsere Python-Umgebung eingerichtet und die erforderlichen Pakete aktualisiert und sind nun bereit, uns Machine Learning in Aktion anzusehen.

Im weiteren Verlauf des Buches werden wir neben dem Machine Learning selbst verschiedene Verfahren zur Vorverarbeitung von Daten vorstellen, die dabei helfen, mit verschiedenen Lernalgorithmen die beste Leistung zu erzielen. Wir werden uns im gesamten Buch ziemlich ausführlich mit Klassifikationsalgorithmen befassen, aber auch einige Verfahren der Regressionsanalyse und des Clusterings betrachten.

Vor uns liegt eine interessante Tour, auf der viele leistungsfähige Verfahren des weiten Felds des Machine Learning zur Sprache kommen. Wir gehen jedoch schrittweise vor und bauen auf das in den einzelnen Kapiteln allmählich erworbene Wissen auf. Im nächsten Kapitel beginnt diese Tour mit der Implementierung einer der ersten Lernalgorithmen zum Zweck der Klassifikation, die uns auf das Kapitel 3 (*Machine-Learning-Klassifikatoren mit scikit-learn verwenden*) vorbereitet, in dem wir die scikit-learn-Bibliothek nutzen werden, um erweiterte Lernalgorithmen zu erörtern.