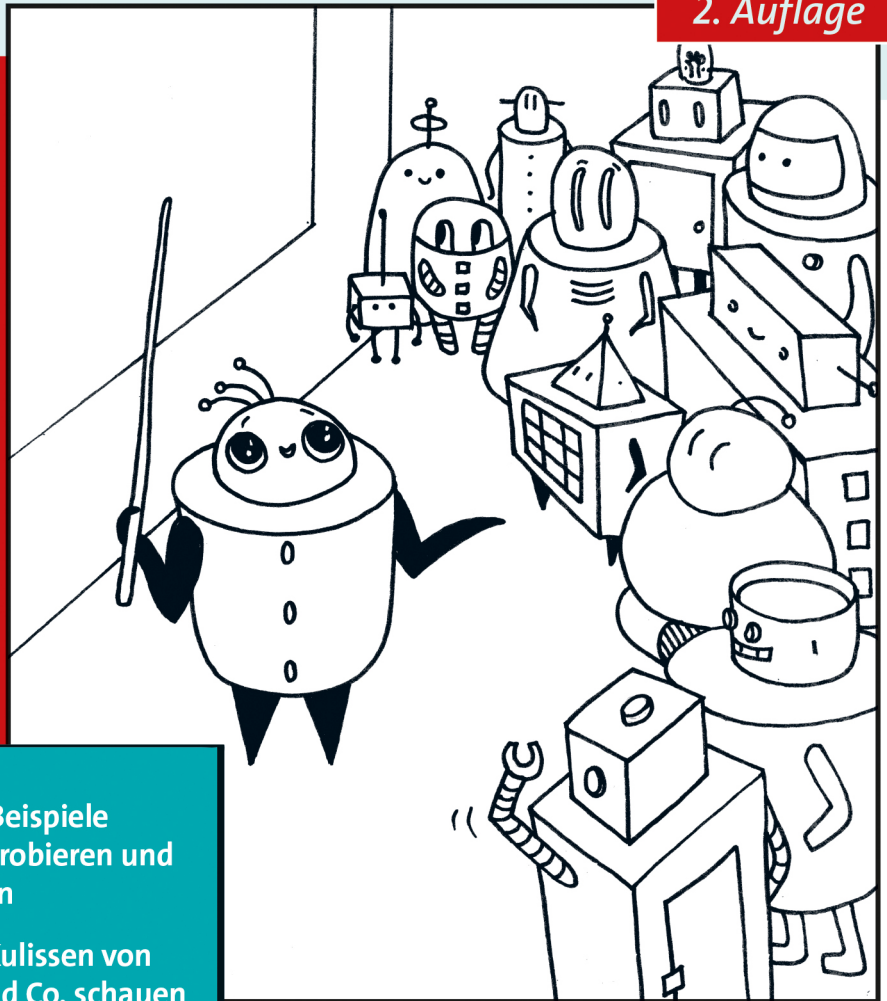


Pit Noack · Sophia Sanner

Künstliche Intelligenz verstehen *Eine spielerische Einführung*

2. Auflage



- + JavaScript-Beispiele online ausprobieren und modifizieren
- + Hinter die Kulissen von ChatGPT und Co. schauen
- + Spiele-KI, Graphen, Neuronale Netze u. v. m.

Kapitel 1

Einleitung

Es fällt nicht schwer, Namen von historischen Persönlichkeiten zu nennen, die besonders intelligent waren. Wir denken spontan an Köpfe wie Albert Einstein, Hannah Arendt, Leonardo da Vinci oder Simone de Beauvoir. Vielleicht haben wir auch einen Freund – nennen wir ihn Karsten –, der herausragend Schach spielt, oder eine Freundin – nennen wir sie Sabine –, die im Studium immer alles als Erste kapiert hat und im Alltag regelmäßig schlaue Lösungen für vertrackte Probleme aus dem Hut zaubert.

Weitaus schwieriger, als intelligente Menschen zu identifizieren, ist es, zu erklären, was Intelligenz eigentlich ist. Eine allgemein anerkannte Definition fehlt. Somit ist es wenig überraschend, dass auch der Begriff *Künstliche Intelligenz* (KI) höchst umstritten ist.

Den herausragenden Schachspieler Karsten werden wir ohne Umschweife intelligent nennen. Aber viele von uns würden zögern, ein Schachprogramm als intelligent zu bezeichnen, selbst wenn dieses weitaus besser spielen kann als unser Freund. Woher kommt diese Zurückhaltung?

Das dürfte unter anderem daran liegen, dass sich viele Menschen generell schwer damit tun, Maschinen menschliche Eigenschaften zuzuschreiben. Ein weiterer und aus unserer Sicht der entscheidende Grund: Alle Anwendungen, die aktuell unter dem Namen »künstliche Intelligenz« firmieren, sind jeweils auf *eine einzige Klasse von Aufgaben* spezialisiert. Versuchen Sie einmal, mit einem Schachprogramm über Literatur zu diskutieren, probieren Sie, eine Bilderkennungssoftware zu einer Partie Halma zu überreden, oder bitten Sie Ihr Navigationssystem, eine Knobelaufgabe zu lösen oder eine Geschichte zu einem vorgegebenen Thema zu improvisieren.



Diese Spezialisierung markiert einen wesentlichen Unterschied zwischen menschlicher Intelligenz und dem, was Computerprogramme aktuell zu leisten imstande sind: Unsere Freundin Sabine vermag sich auf sehr unterschiedliche Probleme und Aufgaben im kürzester Zeit einzustellen. Auch von Freund Karsten dürfen wir erwarten, dass er seine im Schachspiel erworbenen Fähigkeiten zum strategischen und weit vorausschauenden Denken auch auf andere Brettspiele oder gar auf Alltagssituationen übertragen kann.

Zwar können Sprachmodelle wie ChatGPT im Medium der natürlichen Sprache sehr unterschiedliche Aufgaben mehr oder weniger erfolgreich bewältigen. Sie sind aber von einer universell anwendbaren Intelligenz, die wir bei Sabine und Karsten beobachten können, noch weit entfernt. Es ist unter Fachleuten umstritten, ob eine solche *Allgemeine Künstliche Intelligenz* (AKI) jemals erreicht werden kann – und wenn ja, auf welchem Wege wir dorthin gelangen könnten. Wir halten uns in dieser Frage mit einer Einschätzung zurück, sind uns aber in einem Punkt sicher: Es genügt nicht, einen Haufen von spezialisierten Programmen zu versammeln und dann mal das eine, mal das andere aufzurufen, je nachdem, was gerade anliegt.

Fassen wir also zusammen: Die Kritik am Begriff *Künstliche Intelligenz* halten wir für berechtigt. Trotzdem lässt sich kaum bestreiten, dass unter diesem Begriff viele erstaunliche Anwendungen versammelt sind, die unser aller Leben bereits maßgeblich bestimmen und verändern – vielleicht sogar mehr, als uns lieb ist. Ziel dieses Buchs ist es, allen interessierten Leserinnen und Lesern ein grundlegendes Verständnis von KI zu vermitteln, ganz unabhängig davon, welche Vorkenntnisse diese mitbringen.

1.1 Worum es uns in diesem Buch geht

Stellen Sie sich vor, Sie besuchten eine Stadt, in der Sie nie zuvor gewesen sind. Sie planen einen längeren Aufenthalt oder wollen diese Stadt in Zukunft öfter besuchen. Es lohnt sich also, Orientierung zu gewinnen! Wie gehen Sie vor? Werden Sie sämtliche Straßen systematisch von Norden nach Süden und von Osten nach Westen durchkämmen? Werden Sie ausgehend von den Hauptverkehrsadern alle Nebenstraßen durchstreifen oder gar den Stadtplan auswendig lernen? Wohl eher nicht! Wenn Sie eine Stadt erkunden, werden Sie sich vermutlich zunächst einige Orientierungspunkte einprägen: den Bahnhof, das gemütliche Café, das Kunstmuseum, das Restaurant mit dem leckeren chinesischen Essen ... Ausgehend von diesen Punkten werden Sie im Laufe der Zeit die Stadt immer besser kennenlernen.

Dieses Buch nimmt sich das Thema KI in ähnlicher Weise vor. Wir erheben nicht den Anspruch, eine systematische oder gar vollständige Darstellung zu liefern. Vielmehr haben

wir einige Verfahren herausgepickt, die wir besonders lehrreich, interessant oder unterhaltsam finden.

Beim Schreiben dieses Buchs haben wir großen Wert auf anschauliche Beispiele und eine kurzweilige Darstellung gelegt. Um es mit den Worten einer befreundeten Informatikstudentin zu sagen: »An der Uni sind wir in das Thema Q-Learning mit einem Haufen mathematischer Definitionen eingestiegen – euer Buch startet mit einem Eichhörnchen, das seinen Nussvorrat verbummelt hat.«

Sehr viel Mühe haben wir darauf verwendet, die vorgestellten Verfahren praktisch erfahrbar zu machen: Auf der Webseite zum Buch finden Sie zahlreiche Beispielprogramme, die Sie mit nur einem Mausklick im Browser starten können. Die Programme laden ein zum Probieren und Experimentieren und stellen eine direkte Verbindung zwischen Theorie und Praxis her.

1.2 Für wen wir dieses Buch geschrieben haben

Wir sprechen mit diesem Buch einen großen Kreis von Leserinnen und Lesern an. Wenn Sie sich in einer der folgenden Beschreibungen wiederfinden, dann haben wir dieses Buch auch für Sie geschrieben:

- ▶ Sie haben noch nie programmiert und möchten es vielleicht auch nicht lernen, wollen aber mehr über KI erfahren.
- ▶ Sie sind Hobbyprogrammiererin oder Freizeit-Coder und freuen sich über Anregungen für eigene Projekte.
- ▶ Sie stehen am Beginn einer Ausbildung oder eines Studiums in einem Fachgebiet, in dem Sie mit Code und KI in Berührung kommen werden, und möchten sich einen ersten Überblick über KI-Verfahren verschaffen.
- ▶ Sie haben beruflich mit Programmierung zu tun, wissen aber noch wenig über KI und wollen ihr Wissen ausbauen.
- ▶ Sie lehren an einer Schule oder einer Hochschule und sind auf der Suche nach anschaulichen Beispielen für Ihren eigenen Unterricht.

Bei allen Verfahren, die wir vorstellen, war unser erstes Anliegen, ein grundlegendes Verständnis der entscheidenden Ideen zu vermitteln. Wir haben mehr Wert auf ein intuitives Verständnis als auf mathematische Präzision gelegt. Denn wir haben uns gedacht: Ausführliche Darstellungen von technischen Details und Spezialfällen nützen wenig, wenn das Wesentliche unklar bleibt.

Das bedeutet aber nicht, dass Sie irgendetwas verlernen müssen, wenn Sie mehr erfahren und über das Darstellungsniveau dieses Buchs hinausgehen wollen. Wann immer wir einen Sachverhalt grob vereinfacht haben, weisen wir deutlich darauf hin.

1.3 Aufbau der einzelnen Kapitel

Die einzelnen Kapitel sind weitgehend einheitlich aus folgenden Bausteinen zusammengesetzt:

- ▶ Wir stellen eine Klasse von Problemen vor, die wir lösen wollen.
- ▶ Wir erläutern ein Verfahren, das zur Lösung des vorgestellten Problems geeignet ist.
- ▶ Wir präsentieren ein Beispielprogramm, in dem das vorgestellte Verfahren zur Anwendung kommt. Das Programm beschreiben wir zunächst aus der Sicht von Nutzerinnen und Nutzern.
- ▶ Wir stellen den Programmcode vor, der das jeweilige Verfahren umsetzt.
- ▶ Wir geben einen Ausblick auf benachbarte Themen, skizzieren Punkte, die wir weglassen haben, schlagen Modifikationen des Beispielprogramms sowie Codeexperimente vor und geben Denkanstöße.

1.4 Ein Wort an die Programmierunkundigen

Wir richten uns mit diesem Buch ganz bewusst auch an Personen, die noch nie in ihrem Leben programmiert haben. Trotzdem enthält es einige Passagen, in denen wir Code im Detail diskutieren. In diesen Passagen mussten wir die Kenntnis von Grundbegriffen wie *Schleife*, *Funktion*, *Objekt* oder *Klasse* voraussetzen. Falls Ihnen solche Begriffe noch fremd sind, schlagen wir Ihnen eine dieser drei Vorgehensweisen vor:

- ▶ Sie überspringen unsere Codeerläuterungen komplett – das wäre zwar schade, aber zu verschmerzen: Die Passagen, die sich auf Code beziehen, sind für das angestrebte allgemeine Verständnis und die weitere Lektüre nicht zwingend notwendig.
- ▶ Sie überfliegen die Codeabschnitte und beachten dabei vor allem die Tabellen und die Infografiken. Auf diese Weise bekommen Sie immerhin eine Ahnung davon, aus welchen Komponenten der Code besteht und wie diese ineinandergreifen.

- ▶ Wenn Sie es genauer wissen wollen, finden Sie in Anhang A eine knappe Einführung in die verwendete Programmiersprache JavaScript und die p5.js-Bibliothek. Diese Einführung fängt bei null an, setzt also überhaupt keine Vorkenntnisse voraus. Das Studium dieser Einführung sollte Sie in die Lage versetzen, unseren Codeerläuterungen weitgehend zu folgen.

1.5 Beispielprogramme und die Webseite zum Buch

Alle Beispielprogramme können Sie ohne Download, Installation und Anmeldung direkt im Browser starten. Die Links finden Sie auf der Webseite zum Buch:

<https://maschinennah.de/ki-buch>.

The screenshot displays a grid of eight interactive example programs, each with a title, a small illustration, and a 'STARTEN' button. The programs are:

- Kapitel 2**: Texte bauen mit Markow. Illustration: A book character with a face and antennae. Example program: Nonsense-Texter.
- Kapitel 3**: Schreibfehler automatisch korrigieren. Illustration: A grid of numbers with arrows indicating corrections. Example program: Wortvergleich.
- Kapitel 4**: Wörter gruppieren. Illustration: A diagram showing items and transactions. Example program: Assoziationsanalyse.
- Kapitel 5**: Spiele für eine Person lösen. Illustration: A grid of hearts. Example program: Fruchtkräsch.
- Kapitel 6**: Spiele für zwei Personen gewinnen. Illustration: A grid of black and white dots.
- Kapitel 7**: Q-Learning. Illustration: A book character holding a sword.
- Kapitel 8**: K-nächste-Nachbarn. Illustration: A diagram showing connections between nodes. Example program: Tiere erkennen.
- Kapitel 9**: K-means-Clustering. Illustration: A character holding a scale and a bowl. Example program: Wetterdaten gruppieren.

Abbildung 1.1 Die Webseite zum Buch

Profis haben darüber hinaus die Möglichkeit, den Quellcode direkt bei GitHub herunterzuladen: <https://github.com/MaschinenNah/ki-buch>.

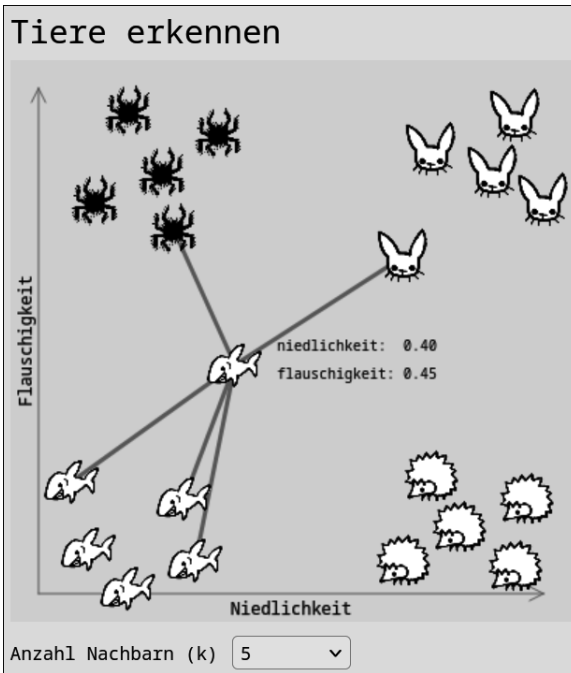


Abbildung 1.2 Das Beispielprogramm »Tiere erkennen«

Experimentierfreudige Leserinnen und Leser können den Code unserer Beispielprogramme eigenhändig modifizieren und das Ergebnis direkt im Browserfenster sehen. Der Online-Editor von p5.js macht es möglich. Abbildung 1.3 zeigt einen Screenshot des Editors: Links sehen Sie den Code, rechts die Ausführung des Programms. Mehr über Nutzung und Funktionen des Online-Editors erfahren Sie in Anhang A in Abschnitt A.3, »Der p5.js-Online-Editor«.

Im Buch werden wir ausschließlich diejenigen Codeabschnitte diskutieren, die für die Umsetzung der vorgestellten KI-Verfahren relevant sind. So stellen wir zum Beispiel in Kapitel 6, »Spiele für zwei Personen gewinnen«, nicht dar, wie wir das Spiel Reversi programmiert haben. Die Funktionsweise der grafischen Benutzeroberfläche, die Repräsentation des Spielfelds oder die Ermittlung der jeweils gültigen Züge besprechen wir nicht. Es geht uns allein darum, die Funktionsweise der KI zu erläutern, die gegen uns Reversi spielt.

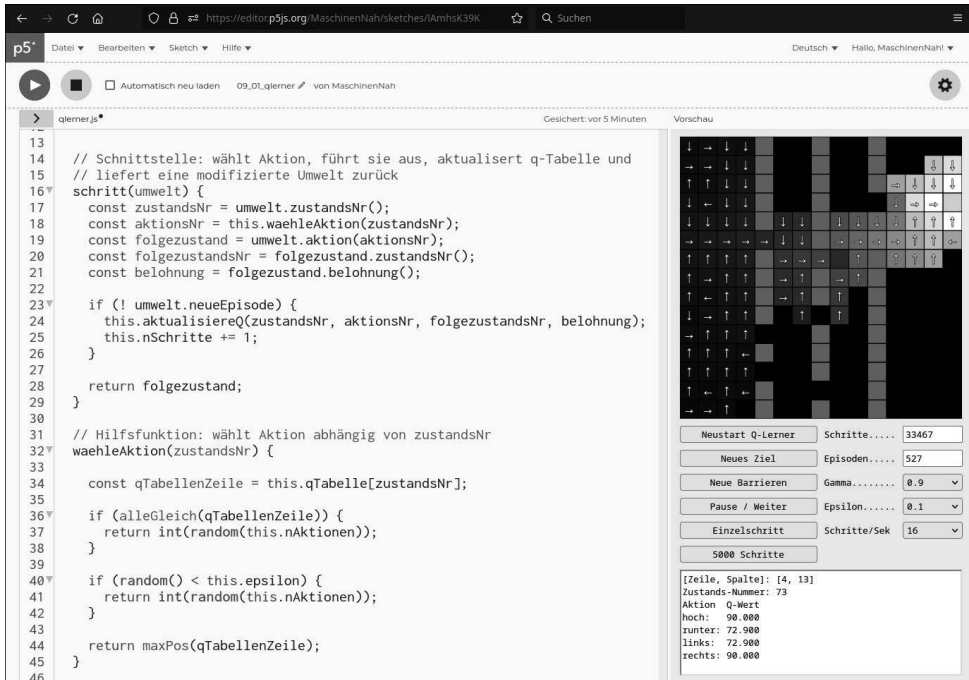


Abbildung 1.3 Im Online-Editor modifizieren Sie auf der linken Seite den Code, auf der rechten Seite sehen Sie das Ergebnis; dafür ist keine Anmeldung notwendig.

Wenn Sie allerdings mehr über das Zusammenspiel der einzelnen Komponenten unserer Beispielprogramme erfahren wollen, stehen Sie nicht mit leeren Händen da: Den online verfügbaren Quellcode haben wir mit erläuternden Kommentaren versehen. Da wir uns mit diesem Buch auch an Anfängerinnen und Einsteiger richten, sind die Kommentare mitunter etwas ausführlicher geraten, als es professionelle Softwareentwicklerinnen und -entwickler für angebracht halten dürften.

1.6 Warum wir JavaScript und p5.js verwendet haben

Programmierprofis werden sich fragen, warum wir für unsere KI-Einführung ausgerechnet JavaScript und p5.js nutzen. Schließlich werden professionelle KI-Anwendungen meist in Python geschrieben. Und in der Tat: Wenn Sie nach der Lektüre dieses Buchs auf den Geschmack gekommen sind und größere KI-Projekte in Angriff nehmen wollen, werden Sie an Python und Python-Umgebungen wie Keras oder PyTorch nicht vorbeikommen.

Das liegt zum einen daran, dass wir waschechte p5.js-Fans sind und diese Sprache seit vielen Jahren erfolgreich in Kursen für jugendliche und erwachsene Programmierneulinge verwenden.

Und auch sonst war unsere Entscheidung didaktisch motiviert: Wir wollen möglichst viele Leserinnen und Leser ansprechen und den Einstieg so unkompliziert wie möglich gestalten. Mit JavaScript, p5.js und dem Online-Editor ist jedes Beispielprogramm nur einen Mausklick entfernt. Die Notwendigkeit eines Downloads, einer Softwareinstallation oder auch nur einer Registrierung würde viele potenziell interessierte Menschen abschrecken.

Zudem programmieren wir in unseren Beispielen vieles »von Hand«, was Keras, PyTorch & Co. als fertige Funktion anbieten. Auch hier gilt: Wenn Sie später auf eine andere Programmiersprache umsteigen, müssen Sie nichts davon verlernen, was Sie in diesem Buch gelernt haben. Im Gegenteil: Die Fachbegriffe, die wir hier vorstellen, bleiben dieselben, ganz unabhängig davon, welche Programmiersprache Sie nutzen. Zudem bieten unsere handprogrammierten Beispiele die Möglichkeit, nachzuvollziehen, was bei Keras und PyTorch hinter den Kulissen passiert.

1.7 Begriffliche Abgrenzung und Fachbegriffe

Wir haben bei der Auswahl der Themen den Begriff *KI* bewusst sehr weit gefasst. Viele der vorgestellten Verfahren werden im strengen Sinne nicht – oder nicht mehr – zum Bereich der *KI* gezählt. So setzen einige Kapitel einen Schwerpunkt auf *Data-Mining*, also auf die Extraktion von Zusammenhängen aus Datenmengen, deren Umfang das menschliche Fassungsvermögen überschreitet. Die Grenzen zwischen *Data-Mining* und *KI* sind aber fließend. Kapitel 9, »K-means-Clustering«, zeigt mit dem k-means-Clustering-Algorithmus ein klassisches *Data-Mining*-Verfahren. Kapitel 12, »Neuronale Netze III: Fehler zurückverfolgen mit dem Neuronentrainer«, stellt die Verbindung zwischen k-means-Clustering und neuronalen Netzen her: Letztlich geht es in beiden Fällen um die Ziehung von Grenzlinien zwischen Datenpunktmengen.

Beim Schreiben dieses Buchs haben wir uns sehr viele Gedanken darüber gemacht, welche Fachbegriffe wir vorstellen und welche wir unerwähnt lassen. Einerseits helfen Fachbegriffe, Ordnung in ein Themengebiet zu bringen und Unterschiede zu markieren. Andererseits kann eine zu große Genauigkeit beim Einstieg eher abschrecken und den Blick auf das Wesentliche verstellen.

1.8 Inhalte, Themen, Kapitel

Im Folgenden skizzieren wir, was Sie in den einzelnen Kapiteln erwartet. Viele der Kapitel können Sie unabhängig voneinander lesen. Wo diese inhaltlich aufeinander aufbauen, haben wir das markiert.

In **Kapitel 2 bis Kapitel 4** stellen wir drei Verfahren vor, die sich besonders gut auf Texte anwenden lassen. Sie stehen in einem losen Zusammenhang, können aber auch unabhängig voneinander gelesen werden:

- ▶ Mittels sogenannter *Markow-Prozesse* lassen sich Texte im Hinblick auf die Häufigkeit bestimmter Buchstaben- und Wortfolgen untersuchen. Anschließend können wir diese Eigenschaften im Sinne einer Parodie reproduzieren, um abhängig vom Ausgangsmaterial etwa deutsch-, englisch- oder französischsprachigen Unsinn zu produzieren. Wir zeigen auch, wie Sie den Markow-Prozess zur Programmierung einer einfachen Textvervollständigung nutzen können, wie wir sie von unseren Smartphones kennen.
- ▶ Der *Levenshtein-Algorithmus* berechnet die Ähnlichkeit zweier Wörter, indem er die kürzeste Schrittfolge ermittelt, die nötig ist, um ein Wort in ein anderes zu verwandeln. Wir nutzen dieses Verfahren zur Programmierung einer einfachen Rechtschreibprüfung, die etwa das Wort »Schampingjong« zu »Champignon« korrigiert.
- ▶ Das Data-Mining-Verfahren *Assoziationsanalyse* findet inhaltliche Zusammenhänge in Playlisten, Warenkörben oder Texten. Unser Beispielprogramm »Begriffsnetz« navigiert durch ein Netz von Assoziationen, die wir aus Sendungsmanuskripten des Deutschlandfunks gewonnen haben. Das Programm lernt etwa, dass die Begriffe »Urknall«, »Weltformel« und »Relativitätstheorie« zusammengehören, ohne eine Idee von Astrophysik oder von überhaupt irgendeinem Thema zu haben.

Kapitel 5 und Kapitel 6 sind der klassischen Spiele-KI gewidmet. Sie bauen inhaltlich aufeinander auf und sollten daher unbedingt hintereinander gelesen werden:

- ▶ Wir haben uns extra für Sie, liebe Leserinnen und Leser, das Ein-Personen-Spiel »Fruchtkräsch« ausgedacht und programmiert. Dabei haben wir uns von Puzzlespielklassikern wie »Candy Crush« oder »Collapse« inspirieren lassen. Wir zeigen, wie eine KI mittels eines Verfahrens namens *Breitensuche* zukünftige Spielzustände vorausrechnen und auf diese Weise den vielversprechendsten Zug ermitteln kann. Sie können Ihre eigenen Fruchtkräsch-Skills direkt mit jenen unserer KI messen.
- ▶ Bei Spielen für zwei Personen reicht es nicht, die denkbaren Spielverläufe per *Breitensuche* zu berechnen, um den jeweils besten Zug zu finden. Die KI muss die den eigenen Interessen entgegengesetzten Absichten einer Gegnerin oder eines Gegners berücksichtigen. Dies leisten die *Tiefensuche* und der *Minimax-Algorithmus*. Im Bei-

spielprogramm können Sie »Reversi« gegen unsere selbst programmierte KI spielen, die Minimax und eine Optimierung namens *Alpha-Beta-Pruning* nutzt.

Kapitel 7 stellt einen grundlegenden Vertreter des *bestärkenden Lernens* vor:

- ▶ *Q-Learning* ist ein automatisches Lernverfahren, bei dem ein Agent (in unserem Fall ein Eichhörnchen) mit einer Umwelt (dem Garten) interagiert, um ein Ziel (das veressene Nussdepot) zu erreichen. Wir führen den Spezialfall des Q-Learnings mittels *Q-Tabelle* im Detail vor.

In **Kapitel 8** und **Kapitel 9** zeigen wir zwei Verfahren, die auf Geometrie beruhen. Sie bauen aufeinander auf und sollten daher gemeinsam gelesen werden:

- ▶ Der *k-nächste-Nachbarn-Algorithmus* ist ein grundlegendes Verfahren zur *Klassifikation* von Daten. Wir lernen, Daten als geometrische Punkte zu interpretieren. Dann können wir nicht klassifizierte *Datenpunkte* einer Klasse zuordnen, indem wir die Entfernungen zu anderen Datenpunkten berechnen, die bereits klassifiziert wurden. Wir führen das praktisch vor mit einer Anwendung, bei der wir anhand wissenschaftlich objektiv gemessener Werte für »Flauschigkeit« und »Niedlichkeit« entscheiden, ob wir es mit einem Häschen, einem Hai, einem Igel oder einer Vogelspinne zu tun haben.
- ▶ Beim *k-means-Clustering-Algorithmus* geht es ebenfalls um die Klassifizierung von Datenpunkten. Allerdings starten wir in diesem Fall mit einer vollständig unklassifizierten Punktmenge. Der Algorithmus ordnet die Datenpunkte in Klassen ein, indem er Gruppen von Punkten findet, die besonders nahe beieinanderstehen. Die praktische Anwendung gruppiert auf diese Weise Messdaten von Wetterstationen.

In **Kapitel 10 bis Kapitel 13** geben wir eine Einführung in die Programmierung *neurona-ler Netze*. Dieser Abschnitt stellt inhaltliche Bezüge zu Kapitel 8 und Kapitel 9 her, die Sie also vorher gelesen haben sollten:

- ▶ Sie lernen, was ein *künstliches Neuron* ist, wie Sie aus Neuronen ein Netz spinnen und wie ein solches *künstliches neuronales Netz* Berechnungen durchführt. Bei dieser Gelegenheit treffen wir das »Häschenproblem« aus Kapitel 8 wieder und lösen es mittels eines neuronalen Netzes.
- ▶ Wir erklären, wie Netze per *überwachtem Lernen* darauf trainiert werden, ein bestimmtes Problem zu lösen, und stellen das *Gradientenabstiegsverfahren* vor, das wir für dieses Training benötigen.
- ▶ Das Beispielprogramm »Neuronentrainer« erlaubt es Ihnen, einem neuronalen Netz beim Lernen zuzuschauen, und lädt zu eigenen Experimenten ein.
- ▶ In Kapitel 13 bieten wir eine flotte Übersicht weiterer Typen und Bauarten neuronaler Netze. Wir erklären Ihnen, was *Faltungsnetze* sind, und führen die Arbeit von *Filterker-*

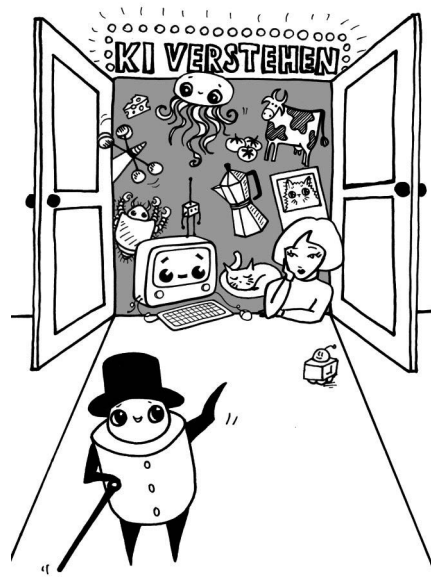
neln anhand eines praktischen Beispiels vor. Zudem skizzieren wir, was es mit *Autoencodern*, *GANs* und *Deep Q-Learning* auf sich hat. Zuletzt kommen wir auf die in dieser Einleitung aufgeworfene Frage nach einer allgemeinen künstlichen Intelligenz zurück.

Das abschließende **Kapitel 14** schlägt die Brücke zu den aktuellsten technischen Entwicklungen.

Es stellt zwei Verfahren vor, die eine unabdingbare Voraussetzung dafür sind, dass *Transformer* wie ChatGPT überhaupt Texte verarbeiten und produzieren können: *Wort- und Sequenzeinbettungen* ordnen Wörter und Sätze so in hochdimensionalen Räumen an, dass sinnhafte und grammatikalische Beziehungen zwischen ihnen als mathematische Relationen berechenbar werden. Es bezieht sich auf fast sämtliche vorangegangene Kapitel und sollte als letztes gelesen werden.

Das **Nachwort** nähert sich dem Thema KI aus einer eher philosophischen und kunstwissenschaftlichen Richtung. Es befasst sich mit der Frage, was es zu bedeuten hat, wenn Computerprogramme Geschichten schreiben, Bilder malen und Musikstücke komponieren.

Jetzt bleibt uns nur noch, Ihnen unterhaltsame und lehrreiche Stunden mit unserem Buch und den Beispielprogrammen zu wünschen. Möge Ihnen das eine oder andere Licht aufgehen! Wenn Sie bei der Lektüre auch nur annähernd so viel Spaß haben, wie uns das Schreiben gebracht hat, und wenn Sie nur einen Bruchteil von dem lernen, was wir beim Schreiben gelernt haben – dann dürfen wir uns sehr glücklich schätzen.



1.9 Dank

An der Entstehung dieses Buchs haben sehr viele Personen mitgewirkt, bei denen wir uns ganz herzlich bedanken:

Stellvertretend für die immer freundlichen und hilfsbereiten Mitarbeiterinnen und Mitarbeiter des Rheinwerk Verlags bedanken wir uns bei der Lektorin Almut Poll. Die Zusammenarbeit mit ihr war eine überaus glückliche Konstellation: Frau Poll hat das Projekt von Anfang an mit Herz und Verstand unterstützt. Es war immer ein Vergnügen, mit ihr didaktische, fachliche und technische Aspekte des Buchs zu diskutieren. Auch für ihre Geduld angesichts unserer künstlerischen Auffassung von Kapitelplanungen und Abgabefristen schulden wir ihr unseren Dank.

Der geschätzte Autorenkollege Torsten T. Will war so freundlich, das komplette Manuskript durchzusehen. Er hat durch seine kompetenten Fragen, Anregungen und Kommentare ganz erheblich zur Verbesserung der Qualität dieses Buchs beigetragen. Marco Rudolph vom Institut für Informationsverarbeitung an der Leibniz Universität Hannover und Lea Reinhart haben weite Teile des Manuskripts gelesen, auf fachliche Korrektheit geprüft, viele wertvolle Hinweise gegeben und Verbesserungsvorschläge gemacht. Torsten, Marco und Lea waren eine unschätzbare Hilfe bei der Realisierung unseres Vorhabens, fachliche Präzision mit anschaulichen Erklärungen zu verbinden. Trotzdem gilt selbstverständlich, dass alle verbleibenden Fehler, Ungereimtheiten und Ungenauigkeiten ganz allein auf unsere Kappe gehen.

Claudia Ley hat in den letzten zwei Jahren einen guten Teil ihrer knappen Freizeit geopfert, um uns bei der Entwicklung der Beispielprogramme und der Texte zu unterstützen. Ihre Geduld und ihr niemals ermüdendes Interesse an dem Projekt haben geholfen, so manche Durststrecke zu überwinden.

Herbert Schmid hat uns bei der Konzeption des Buchs beraten und uns die eine oder andere Perle aus seinem enzyklopädischen IT-Wissen zukommen lassen.

Der Grafiker Ralf Rohde hat auf den letzten Metern bei der Korrektur und Ergänzung von Infografiken sehr wertvolle Dienste geleistet.

Pit Noack bedankt sich zudem bei der Illustratorin Sophia Sanner für die überaus angenehme Zusammenarbeit und bei Lea Reinhart für die Mitarbeit an Kapitel 7 und dem Glossar. Er dankt Stefan Dissmer, Karsten Fecht, Sibylle Feldmann, Vincent Franke, Lukas Greipel, Heiko Idensen, Pina Merkert, Tobias Prinz, Jonas Rahm und Martin Reche – ihr wisst schon, wofür – sowie allen Teilnehmerinnen und Teilnehmern seiner Programmierkurse, vor allem Karin Gaerner und Olivia, Patrik, Rebecca und Tarik Ley.

Kapitel 8

K-nächste-Nachbarn

Eine Tafel Schokolade hat mehr gemeinsame Eigenschaften mit einem Stück Torte als mit einer Karotte. Eine Tomate wiederum hat viel mehr Gemeinsamkeiten mit der Karotte als mit Schokolade. In diesem Kapitel zeigen wir, wie Sie solche Ähnlichkeiten und Nachbarschaftsverhältnisse in Zahlen fassen und diese nutzen können, um Ordnung in große Datenbestände zu bringen und Zusammengehörigkeiten darin zu erkennen.

Worum es in diesem Kapitel geht

- ▶ Zahlenförmige Daten lassen sich durch *Datenpunkte* darstellen.
- ▶ Distanzen zwischen Datenpunkten können wir mit einem einfachen geometrischen Verfahren berechnen.
- ▶ Der *k-nächste-Nachbarn-Algorithmus* nutzt solche Distanzmessungen, um Datenpunkte zu klassifizieren.



Angenommen, ein Datenbestand versammelt Nährwertangaben, die jeweils zwei Werte enthalten, einen für den Zucker- und einen für den Fettgehalt. Dann kann es sehr hilfreich sein, jedes einzelne dieser Wertepaare als einen *Datenpunkt* auf eine Fläche zu zeichnen. Dabei können wir den Zuckergehalt als Position des Punkts in der Waagerechten und den Fettgehalt als Position des Punkts in der Senkrechten nehmen.

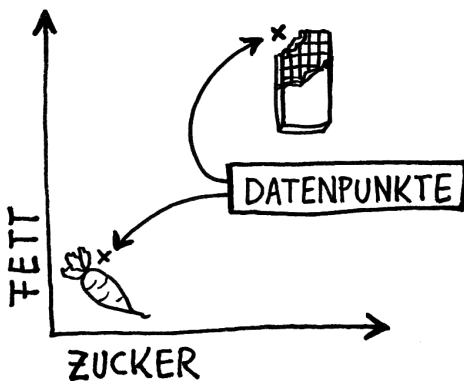


Abbildung 8.1 Messwerte lassen sich als Punkte darstellen.

Ein riesiger Vorteil dieser Darstellung: *Ähnlichkeit* zwischen Datenpunkten lässt sich jetzt *als räumliche Nähe* ablesen und berechnen: Je näher sich zwei Punkte auf der Fläche sind, umso so größer ist die Ähnlichkeit:

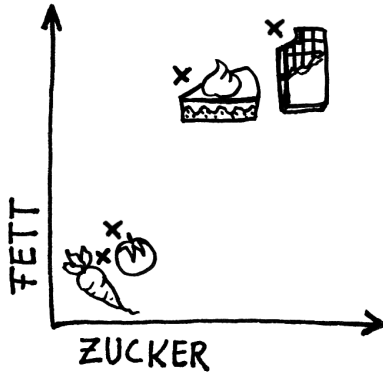


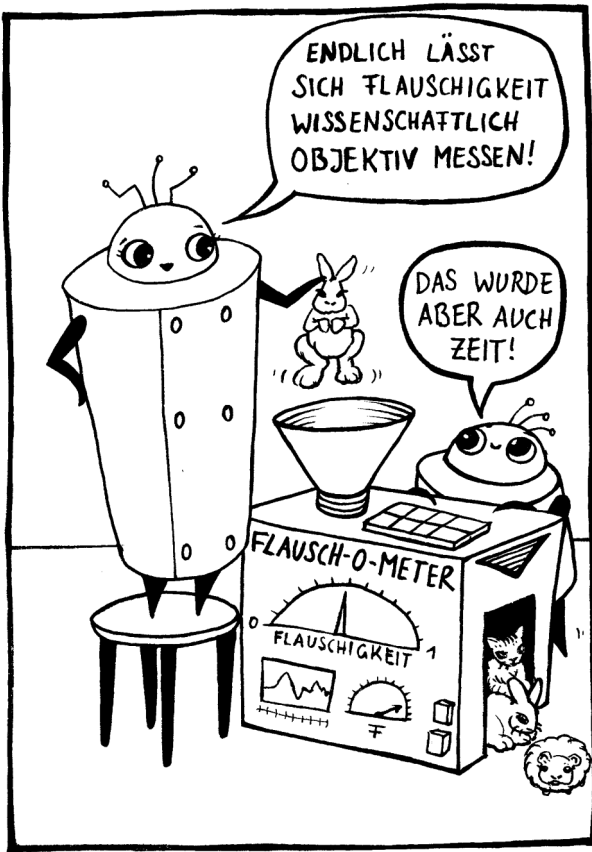
Abbildung 8.2 Räumlich nahegelegene Punkte haben ähnliche Eigenschaften.

Für dieses Kapitel haben wir uns ein alltagsnahes Anwendungsbeispiel ausgedacht, das insbesondere Tierfreundinnen und -freunden anschaulich sein wird: Es geht um die Identifizierung von Tierarten anhand der Eigenschaften *Niedlichkeit* und *Flauschigkeit*.

8.1 Häschen, Igel, Vogelspinne oder Hai?

Das Beispiel geht von folgenden Annahmen aus:

- ▶ Sie haben zwei Messgeräte konstruiert, mit denen Sie die Eigenschaften *Flauschigkeit* und *Niedlichkeit* jeweils auf einer Skala von 0 bis 1 präzise messen können. Das Messverfahren ist selbstverständlich ganz und gar harmlos, es kitzelt höchstens ein wenig.
- ▶ Mittels dieser Geräte haben Sie jeweils fünf Tiere der Spezies Vogelspinne, Häschen, Hai und Igel vermessen und entsprechende Datenpunkte erzeugt: {niedlichkeit: 0.90, flauschigkeit: 0.90, spezies: "haeschen"}. Sie besitzen also einen Datenbestand mit 20 Datenpunkten.
- ▶ Es gibt Tiere, von denen Sie nicht wissen, zu welcher Spezies sie gehören. Anhand der Werte für *Flauschigkeit* und *Niedlichkeit* wollen Sie diese einer der vier Ihnen bekannten Spezies zuordnen. Für diese Zuordnung wollen Sie die räumliche Nähe zu bereits klassifizierten Datenpunkten nutzen.



8.2 Das Beispielprogramm Tiere erkennen

Das Programm »Tiere erkennen« finden Sie wie alle anderen hier besprochenen Beispielprogramme auf der Webseite zum Buch: <https://maschinennah.de/ki-buch>. Wenn Sie das Programm starten, sehen Sie die vorliegenden 20 Datenpunkte und deren Klassifizierung, eingetragen im Koordinatensystem. Die waagerechte Achse steht für *Niedlichkeit*, die senkrechte Achse für *Flauschigkeit*. Sie können an dieser grafischen Darstellung ihres Datenbestands Folgendes ablesen:

- ▶ Vogelspinnen sind flauschig, aber nicht besonders niedlich.
- ▶ Häschen zeichnen sich durch hohe Flauschigkeit und Niedlichkeit aus.

- ▶ Haie sind weder besonders flauschig noch allzu niedlich.
- ▶ Igel wiederum sind außerordentlich niedlich, eine ausgeprägte Flauschigkeit hingegen gehört nicht zu ihren hervorstechenden Eigenschaften.

Wir müssen hier den Einwand gelten lassen, dass in unserer Darstellung auch die Haie recht niedlich geraten sind. Zudem gibt es nachweislich Personen, die sogar Vogelspinnen süß finden.

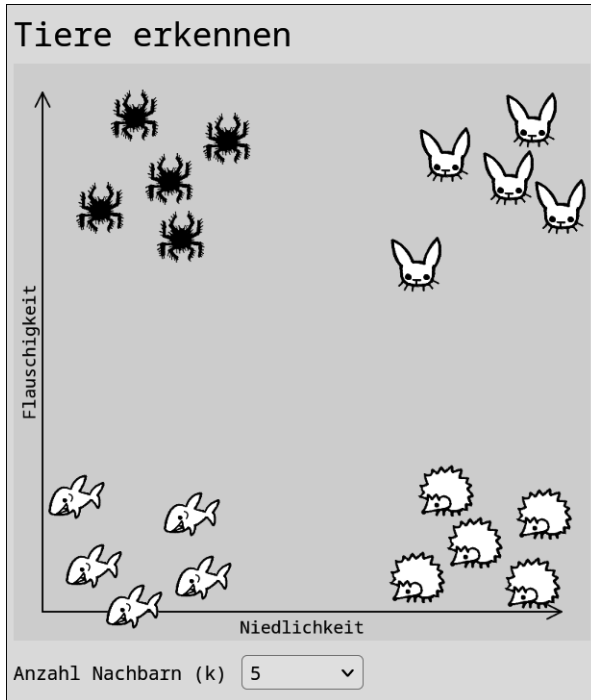


Abbildung 8.3 Das Beispielprogramm »Tiere erkennen«: Die 20 bekannten Datenpunkte sind hier abhängig von den Parametern »Niedlichkeit« und »Flauschigkeit« im Koordinatensystem angeordnet.

Mit dem Drop-down-Menü ANZAHL NACHBARN legen Sie fest, wie viele nächste Nachbarn zur Einordnung eines nicht klassifizierten Datenpunkts genutzt werden sollen.

Die Erzeugung eines neuen Datenpunkts ist ganz einfach: Klicken Sie mit der Maus auf eine beliebige Stelle im Koordinatensystem. Das Programm ordnet den neuen Datenpunkt sogleich einer Spezies zu. Die Zuordnung passiert abhängig davon, welcher Spezies die meisten der benachbarten Datenpunkte angehören. Abbildung 8.4, Abbildung 8.5 und Abbildung 8.6 zeigen Beispiele für solche Zuordnungen.

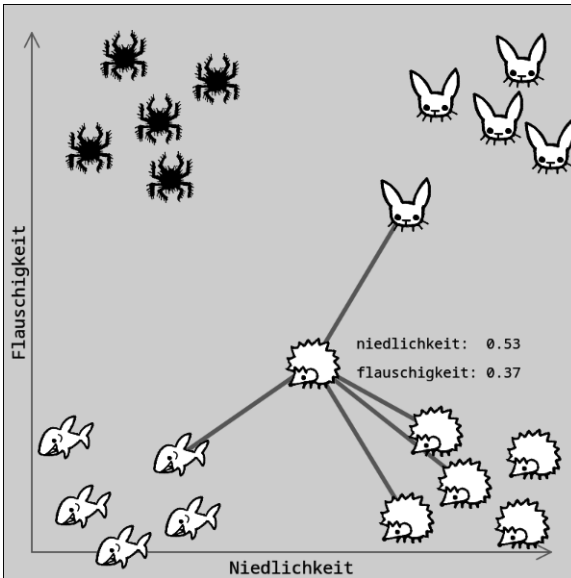


Abbildung 8.4 Das Programm ordnet unbekannte Datenpunkte den vier Spezies Vogelspinne, Häschen, Hai und Igel zu – hier wurde ein Igel erkannt, weil drei der fünf nächsten Nachbarn Igel sind.

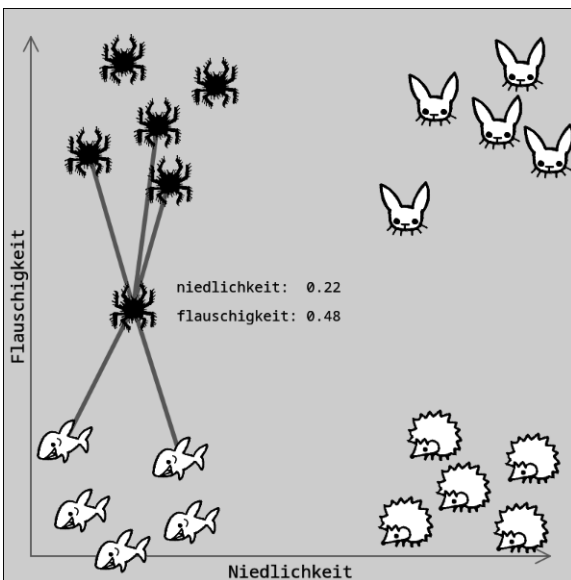


Abbildung 8.5 In diesem Fall hat das Programm eine Vogelspinne identifiziert, weil drei der fünf nächsten Nachbarn zu dieser Spezies gehören.

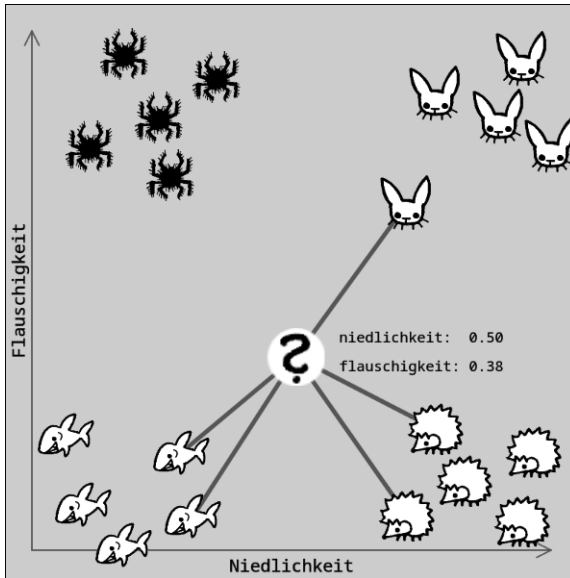


Abbildung 8.6 Weil es keinen eindeutigen »Sieger« gibt, hält sich das Programm mit einer Beurteilung zurück.

8.3 Entfernungen bestimmen mit Pythagoras

Jetzt kennen Sie das Prinzip des k-nächste-Nachbarn-Algorithmus. Es bleibt zu klären, wie wir die Entfernung zwischen zwei Datenpunkten bestimmen. Hier hilft uns ein alter Bekannter aus dem Matheunterricht weiter – der *Satz des Pythagoras*. Diese Gleichung macht eine einfache und nützliche Aussage über die Längenverhältnisse der Seiten eines rechtwinkligen Dreiecks. Dabei ist c die längste Seite, a und b die beiden kürzeren Seiten:

$$a^2 + b^2 = c^2$$

Abbildung 8.7 zeigt, wie Sie durch eine einfache Umformung dieser Gleichung die Länge von c berechnen können, wenn Ihnen die Längen von a und b bekannt sind.

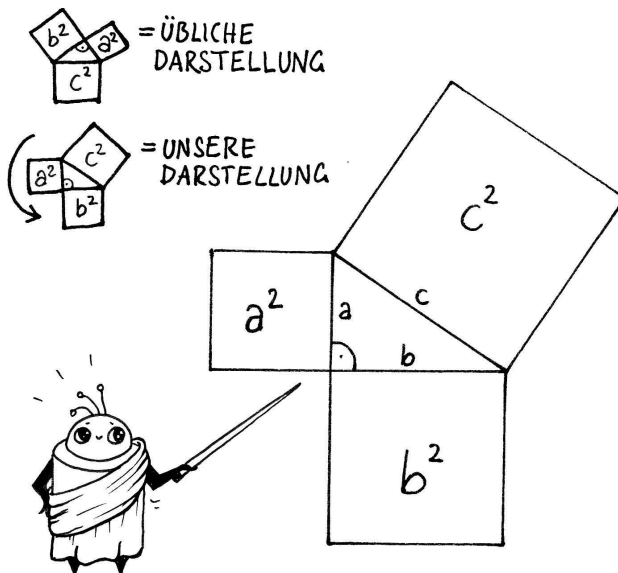
In der Abbildung haben wir das Dreieck abweichend von der üblichen Darstellung des Pythagoras-Satzes so gedreht, dass der rechte Winkel des Dreiecks unten links steht. Damit wollen wir deutlich machen, dass wir die Formel zur Berechnung der Länge von c auch

nutzen können, um die Entfernung zwischen zwei Punkten P1 und P2 im Koordinatensystem zu ermitteln: Wir benennen die Seiten a , b und c schlichtweg um:

- ▶ Aus a wird die Distanz der beiden Punkte auf der x-Achse: Δx .
- ▶ Aus b wird die Distanz der beiden Punkte auf der y-Achse: Δy .
- ▶ Aus c wird die Distanz zwischen den beiden Punkten P1 und P2.

Das Zeichen Δ ist der griechische Buchstabe Delta, der in Formeln und Gleichungen häufig für Differenzen steht.

DER GUTE ALTE SATZ DES PYTHAGORAS



BERECHNUNG DER LÄNGE VON c :

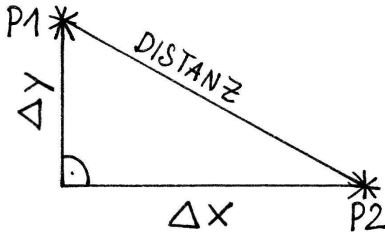
$$a^2 + b^2 = c^2$$

$$c^2 = a^2 + b^2 \quad \left. \begin{array}{l} \text{SEITEN} \\ \text{VERTAUSCHEN} \end{array} \right\}$$

$$c = \sqrt{a^2 + b^2} \quad \left. \begin{array}{l} \text{WURZEL} \\ \text{ZIEHEN} \end{array} \right\}$$

Abbildung 8.7 Mit dem Satz des Pythagoras können wir die Länge der Seite c berechnen ...

BERECHNUNG DER DISTANZ
ZWISCHEN ZWEI PUNKTEN P1 UND P2



$$\begin{aligned}\Delta x &= x \text{ VON } P_1 - x \text{ VON } P_2 \\ &= \text{DISTANZ IN } x\text{-RICHTUNG}\end{aligned}$$

$$\begin{aligned}\Delta y &= y \text{ VON } P_1 - y \text{ VON } P_2 \\ &= \text{DISTANZ IN } y\text{-RICHTUNG}\end{aligned}$$

DISTANZ ZWISCHEN P1 UND P2
MIT PYTHAGORAS =

$$\sqrt{\Delta x^2 + \Delta y^2}$$

Δ ist der griechische Buchstabe Delta

Abbildung 8.8 ... und dies für die Ermittlung der Distanz zwischen zwei Punkten nutzen.

ii

Hintergrund: Euklidische und nicht-euklidische Geometrien

Die hier beschriebene, mit dem Satz des Pythagoras berechnete Entfernung zwischen zwei Punkten wird auch *euklidische Distanz* genannt. Dieser Name bezieht sich auf die vom griechischen Mathematiker Euklid begründete *euklidische Geometrie*. Sie beschreibt einen Raum, in dem Aussagen gelten wie: »Die Summe der Winkel in einem Dreieck beträgt immer 180 Grad« oder »Zwei parallele Linien berühren sich in keinem Punkt«.

Ab dem 19. Jahrhundert haben Mathematiker wie Felix Klein und Henri Poincaré begonnen, alternative, also *nicht-euklidische Geometrien* zu untersuchen. Ein einfaches Beispiel ist die Geometrie auf der Kugeloberfläche. Hier haben die eben genannten Aussagen zur Winkelsumme im Dreieck und den Parallelen keine Gültigkeit.

Jenseits solcher anschaulichen Beispiele untersuchen Mathematikerinnen und Mathematiker bis heute, unabhängig von jeder Anschauung oder Erfahrung, *wie überhaupt Räume beschaffen und Dinge zueinander angeordnet sein können*. Gibt es unendlich viele Möglichkeiten, unendlich viele Geometrien und Topologien? Lassen sich diese sinnvoll in Gruppen unterteilen?

Das sind höchst spannende Fragen, die sehr viel mehr mit den Thema dieses Buches zu tun haben, als auf den ersten Blick offensichtlich sein mag. Wir sprechen im Kontext von KI nämlich unentwegt über Räume, Dimensionen, Anordnungen von Objekten, Zustandsübergänge, Nachbarschaften, Entfernungen und deren Berechnung.

8.4 Der Code im Detail

Im Code unseres Beispielprogramms sind Objekte der Klasse `KNN` für die Umsetzung des **k-nächste-Nachbarn-Algorithmus** zuständig. Sie besitzt eine Eigenschaft und vier Funktionen, die in Tabelle 8.1 aufgelistet sind.

Name	Aufgabe
<code>datenpunkte</code>	Ein Array mit den bereits klassifizierten Datenpunkten in diesem Format: <pre>{niedlichkeit: 0.90, flauschigkeit: 0.90, spezies: "haeschen"}</pre>
<code>klassifiziere(datenpunktA, k)</code>	Klassifiziert einen Datenpunkt und liefert diesen zurück. <code>k</code> bestimmt die Anzahl der Nachbarn, die zur Klassifizierung genutzt werden. Ruft die Funktionen <code>nachbarn()</code> und <code>zaehlung()</code> auf.
<code>nachbarn(datenpunktA, k)</code>	Liefert ein Array mit den <code>k</code> nächsten Nachbarn von <code>datenpunktA</code> . Ruft die Funktion <code>distanz()</code> auf.

Tabelle 8.1 Eine Eigenschaft und vier Funktionen der Klasse `KNN`

Name	Aufgabe
distanz(datenpunktA, datenpunktB)	Ermittelt die Distanz zwischen datenpunktA und datenpunktB.
zaehlung(nachbarn)	Zählt die Häufigkeit der einzelnen Spezies in dem Datenpunkt-Array nachbarn. Rückgabewert ist eine Rangfolge im folgenden Format: [["igel", 3], ["hai", 1], ["haeschen", 1]]

Tabelle 8.1 Eine Eigenschaft und vier Funktionen der Klasse KNN (Forts.)

Listing 8.1 zeigt die Funktion zur Berechnung der Distanz zwischen zwei Datenpunkten, so wie wir sie in Abbildung 8.8 dargestellt haben. `deltaX` und `deltaY` sind die Distanzen zwischen den beiden Punkten in *x*- und *y*-Richtung. Die beiden Deltas werden anschließend quadriert und summiert. Die Quadratwurzel des Ergebnisses ist die Distanz:

```
distanz(datenpunktA, datenpunktB) {
  const deltaX = datenpunktA.niedlichkeit - datenpunktB.niedlichkeit;
  const deltaY = datenpunktA.flauschigkeit - datenpunktB.flauschigkeit;
  return sqrt(deltaX * deltaX + deltaY * deltaY);
}
```

Listing 8.1 Die Funktion `knn.distanz()` berechnet die Entfernung zwischen zwei Datenpunkten.

Die Funktion `nachbarn()` liefert die *k* nächsten Nachbarn zu einem gegebenen `datenpunktA` – im Fall unserer Anwendung ist `datenpunktA` ein noch nicht klassifizierter Punkt. Um die Nachbarn zu finden, sortiert `sort()` die `datenpunkte` abhängig von der Distanz zu `datenpunktA`. Anschließend wählt `slice()` die ersten *k* Datenpunkte aus:

```
nachbarn(datenpunktA, k) {
  this.datenpunkte.sort((L, R) =>
    this.distanz(datenpunktA, L) - this.distanz(datenpunktA, R));
  return datenpunkte.slice(0, k);
}
```

Listing 8.2 Die Funktion `knn.nachbarn()` liefert benachbarte Datenpunkte.

Jetzt fehlt noch die Auswertung des Rückgabewerts von `nachbarn()`: Sind die meisten der benachbarten Datenpunkte Vogelspinnen, Häschen, Haie oder Igel? Diese Auswertung liefert die Funktion `zaehlung()`.

Eine `for`-Schleife durchläuft alle im Argument übergebenen Datenpunkte und zieht jeweils die `spezies` heraus. Ist diese `spezies` im Objekt `zaehlung` noch nicht als Schlüssel vorhanden, wird der Schlüssel angelegt und mit dem Wert 1 initialisiert. Andernfalls wird der zu diesem Schlüssel eingetragene Wert um 1 erhöht. Dies ähnelt übrigens dem Vorgehen, das wir in Abschnitt 2.2, »Der Code des Nonsense-Texters unter der Lupe«, angewendet haben, als es um die Zählung von Worthäufigkeiten im Rahmen eines Markow-Prozesses ging. Abschließend wird das Objekt `zaehlung` per `Object.entries()` in ein Array umgewandelt. Die Funktion `sort()` sortiert das Array nach Anzahl. Das sortierte Array ist der Rückgabewert.

```

zaehlung(nachbarn) {
  const zaehlung = {};

  for (let datenpunkt of nachbarn) {
    const spezies = datenpunkt.spezies;
    if (!zaehlung[spezies]) {
      zaehlung[spezies] = 1;
    } else {
      zaehlung[spezies] += 1;
    }
  }

  const alsArray = Object.entries(zaehlung);
  alsArray.sort((L, R) => R[1] - L[1]);
  return alsArray;
}

```

Listing 8.3 Die Funktion `knn.zaehlung()` liefert eine Rangfolge der Spezies der in der Nachbarschaft gefundenen Datenpunkte.

In der Funktion `klassifiziere()` greifen alle vorgestellten Komponenten ineinander. Die Funktion erwartet als erstes Argument einen nicht klassifizierten `datenpunktA`. Das zweite Argument `k` bestimmt die Anzahl der Nachbarn, die für die Klassifikation relevant sind.

Die eben vorgestellten Funktionen liefern die entsprechenden nächsten Nachbarn und deren Auswertung nach Häufigkeit. Dann bleibt nur noch eine Sache zu tun, nämlich, die Auswertung in eine erkannte Spezies zu übersetzen – oder eben nicht:

- ▶ Die Bedingung `zaehlung.length == 1` ist genau dann erfüllt, wenn alle Nachbarn derselben Spezies angehören. In diesem Fall ist die Zuordnung eindeutig.
- ▶ Ein wenig komplizierter wird es, wenn die Auswertung mehrere Einträge enthält. Ist die Häufigkeit des ersten Eintrags größer als die des zweiten, dann ist die Zuordnung ebenfalls klar: Die Spezies des vorne stehenden Eintrags wurde identifiziert.
- ▶ Wenn keine dieser beiden Bedingungen erfüllt ist, gibt es keinen klaren Spitzenreiter: Die Spezies von `datenpunktA` erhält den Wert "unbekannt".

```

klassifiziere(datenpunktA, k) {
  const nachbarn = this.nachbarn(datenpunktA, k);
  const zaehlung = this.zaehlung(nachbarn);

  if (zaehlung.length == 1 || zaehlung[0][1] > zaehlung[1][1]) {
    datenpunktA.spezies = zaehlung[0][0];
  } else {
    datenpunktA.spezies = "unbekannt";
  }
  return datenpunktA;
}

```

Listing 8.4 Die Funktion `knn.klassifiziere()` ordnet den unbekanntem `datenpunktA` einer Spezies zu.

8.5 Ideen zum Weitermachen

Der k-nächste-Nachbarn-Algorithmus ist nicht besonders anspruchsvoll. Es sollte nicht schwer sein, unser Beispielprogramm zu modifizieren.

Sie könnten etwa eine weitere Tierart in das Programm aufnehmen. Hierfür sind lediglich zwei Ergänzungen notwendig:

- ▶ Zeichnen Sie das Tier und speichern Sie die Zeichnung im Ordner `daten` unter dem entsprechenden Namen. Ideal ist hier das PNG-Format mit einem Alpha-Kanal, also einem durchsichtigen Hintergrund.
- ▶ Schreiben Sie in die Datei `daten/datenpunkte.js` einige Datenpunkte, die Vertreter der neu hinzugefügten Tierart repräsentieren.

Alternativ könnten Sie Tiere auch nach anderen Eigenschaften ordnen: Wie wäre es, wenn sie Tiere nach Gewicht und Zutraulichkeit charakterisierten? Sie könnten auch ein ganz anderes Thema bearbeiten, etwa indem Sie Nahrungsmittel nach ihrem Fett- und

Zuckergehalt klassifizieren. Überlegen Sie dabei, welche Eigenschaften sich überhaupt sinnvoll auf einer Skala eintragen lassen.



8.6 Zusammenfassung und Ausblick

Der k-nächste-Nachbarn-Algorithmus ist ein einfaches Verfahren, um zahlenförmige Daten zu klassifizieren. Der Cartoon mit der falsch klassifizierten Katze zeigt die Grenzen auf: Es können nur Klassen erkannt werden, die schon vorher bekannt waren. Vielen ist die Erfahrung geläufig, von einem Computerprogramm oder einem formalen Prozess in ein Schema hineingepresst zu werden, das offensichtlich nicht passt.

Die in Abschnitt 8.3, »Entfernungen bestimmen mit Pythagoras«, vorgestellte Formel eignet sich ebenfalls für Daten mit mehr als zwei Dimensionen. Auch wenn wir uns das bildlich nur schwer vorstellen können: Wenn die Nährwertangaben eines Fertiggerichts jeweils Werte für Kohlenhydrate, Fett, Eiweiß, Ballaststoffe und Salz enthalten, können wir jedes Fertiggericht als Datenpunkt in einem fünfdimensionalen Raum behandeln und damit rechnen. Die Formel zur Berechnung der Entfernung wird nach demselben Prinzip aufgebaut. Wird die Anzahl der Dimensionen allerdings sehr groß, eignet sich dieses Berechnungsverfahren nicht mehr. Das liegt am sogenannten Fluch der Dimensionalität, den wir in Abschnitt 14.5.6 erklären. Dort stellen wir auch ein alternatives Verfahren zur Entfernungsmessung vor, die *Kosinus-Ähnlichkeit*.

Inhalt

Materialien zum Buch	16
Vorwort zur zweiten Auflage	17

1 Einleitung 19

1.1 Worum es uns in diesem Buch geht	20
1.2 Für wen wir dieses Buch geschrieben haben	21
1.3 Aufbau der einzelnen Kapitel	22
1.4 Ein Wort an die Programmierunkundigen	22
1.5 Beispielprogramme und die Webseite zum Buch	23
1.6 Warum wir JavaScript und p5.js verwendet haben	25
1.7 Begriffliche Abgrenzung und Fachbegriffe	26
1.8 Inhalte, Themen, Kapitel	27
1.9 Dank	30

2 Texte bauen mit Markow 31

2.1 Das Beispielprogramm Nonsense-Texter	35
2.2 Der Code des Nonsense-Texters unter der Lupe	37
2.2.1 Die Komponenten des Markow-Objekts	38
2.2.2 Übergänge lernen	38
2.2.3 Nonsense-Texte produzieren	40
2.3 Das Beispielprogramm Wörter vorschlagen	43
2.3.1 Die Komponenten des Markow-Objekts	43
2.3.2 Übergänge und Häufigkeiten lernen	44
2.4 Wörter vorschlagen	47
2.5 Gewichteter Zufall	48
2.6 Ideen zum Weitermachen	50
2.7 Zusammenfassung und Ausblick	51

3	Schreibfehler automatisch korrigieren	53
3.1	Das Beispielprogramm Wortvergleich	54
3.2	Die Matrix befüllen	57
3.2.1	Die Füllung der oberen Zeile und der linken Spalte	57
3.2.2	Die Füllung der verbleibenden Zellen	58
3.2.3	Die drei Schritte unter der Lupe	62
3.3	Die Umsetzung im Beispielprogramm	62
3.3.1	Das Levenshtein-Objekt	62
3.3.2	Die Funktion matrix()	63
3.4	Das Beispielprogramm Korrekturvorschläge	65
3.5	Ideen zum Weitermachen	67
3.6	Zusammenfassung und Ausblick	68
4	Wörter gruppieren	69
4.1	Items und Transaktionen	71
4.2	Kenngrößen der Assoziationsanalyse	72
4.2.1	Support	72
4.2.2	Confidence	73
4.2.3	Lift	74
4.3	Ein Beispiel von Hand gerechnet	76
4.4	Das Beispielprogramm Begriffsnetz	79
4.4.1	Die Datenquelle	81
4.4.2	Beschränkung der Anwendung auf Item-Paare	81
4.5	Eine Tour durch den Code	82
4.5.1	Die Klasse Begriffsnetz	82
4.5.2	Enkodierung der Transaktionen	83
4.5.3	Befüllung der Arrays für Support	85
4.5.4	Befüllung der Arrays für Confidence und Lift	87
4.5.5	Die Funktion assoziationen()	87
4.6	Ideen zum Weitermachen	88

4.7	Zusammenfassung und Ausblick	90
4.7.1	Der Apriori-Algorithmus	90
4.7.2	Transaktionstabellen	90
4.7.3	Eine Übersicht aller Fachbegriffe aus diesem Kapitel	91

5 Spiele für eine Person lösen 93

5.1	Das Spiel Fruchtkräsch	93
5.2	Wie findet die KI den besten Zug?	95
5.3	Eine vielseitig einsetzbare Spiel-KI	98
5.4	Die Klasse Spielzustand	99
5.4.1	Züge liefern modifizierte Spielzustände	100
5.4.2	Die möglichen Züge	100
5.4.3	Die Bewertung eines Spielzustands	100
5.4.4	Ein Gedächtnis für Züge	101
5.4.5	Die Schnittstelle im Überblick	101
5.5	Die Klasse KI	102
5.5.1	Alle Folgezustände eines Spielzustands berechnen	102
5.5.2	Alle Spielverläufe per Warteschlangenverfahren berechnen	103
5.5.3	Die Spielzustände nach Bewertung sortieren	106
5.5.4	Die Funktion <code>besterZug()</code>	106
5.5.5	Die Funktionen des KI-Objekts im Überblick	107
5.6	Ideen zum Weitermachen	107
5.7	Zusammenfassung und Ausblick	108

6 Spiele für zwei Personen gewinnen 109

6.1	Das Spiel Reversi	110
6.2	Das Beispielprogramm Reversi KI	111
6.3	Der Minimax-Algorithmus	112
6.3.1	Anwendungsgebiete und Grenzen des Minimax-Algorithmus	114

6.4	Tiefensuche und Rekursion	115
6.4.1	Breitensuche und Tiefensuche	116
6.4.2	Die Paradoxie der Rekursion	117
6.4.3	Verzweigte Rekursion	121
6.5	Die Klasse Spielzustand	123
6.5.1	Die Bewertungsfunktion	123
6.5.2	Die Schnittstelle im Überblick	125
6.6	Die Klasse KI	126
6.7	Beschleunigung mit Alpha-Beta-Pruning	130
6.8	Ideen zum Weitermachen	131
6.9	Zusammenfassung und Ausblick	132
7	Q-Learning	133
<hr/>		
7.1	Das Eichhörnchen und das Nussversteck	134
7.2	Umwelt, Agent, Aktion und Belohnung	139
7.2.1	Das Verhältnis von Agent und Umwelt	140
7.3	Die Q-Tabelle	141
7.3.1	Q steht für Qualität	142
7.4	Das Beispielprogramm Q-Lerner	142
7.5	Die Q-Tabelle befüllen	147
7.5.1	Warum funktioniert das?	149
7.6	Der Code unter der Lupe	150
7.6.1	Die Umwelt	150
7.6.2	Der Q-Lerner	151
7.7	Gamma bestimmt die Weitsicht	152
7.8	Epsilon: Erforschung oder Anwendung	154
7.9	Ein zweiter Blick auf den Code	156
7.10	Alpha	158
7.11	Was wir weggelassen haben	159
7.11.1	Komplexere Umwelten	160
7.11.2	Kosten für Aktionen	160

7.11.3	Belohnungen mit Zustands-Aktions-Paaren verknüpfen	160
7.11.4	Mehrere Belohnungszustände	160
7.12	Ideen zum Weitermachen	161
7.12.1	OpenAI Gym	162
7.12.2	Das Buch von Sutton und Barto	163
7.13	Zusammenfassung und Ausblick	163
7.13.1	Menschliches Lernen vs. Q-Learning	163
7.13.2	Die Grenzen des Verfahrens	164
8	K-nächste-Nachbarn	167
<hr/>		
8.1	Häschen, Igel, Vogelspinne oder Hai?	168
8.2	Das Beispielprogramm Tiere erkennen	169
8.3	Entfernungen bestimmen mit Pythagoras	172
8.4	Der Code im Detail	175
8.5	Ideen zum Weitermachen	178
8.6	Zusammenfassung und Ausblick	179
9	K-means-Clustering	181
<hr/>		
9.1	Clusterbildung in Aktion	183
9.1.1	Mittelwert, Zentrum, Schwerpunkt	183
9.1.2	Die Schrittfolge des k-means-Clustering-Algorithmus	185
9.2	Das Beispielprogramm Wetterdaten gruppieren	186
9.3	Der Code	188
9.3.1	Zentren zufällig setzen	189
9.3.2	Datenpunkte zuordnen	189
9.3.3	Zentren neu berechnen	190
9.4	Grenzen des Verfahrens	191
9.4.1	Unsinnige Gruppierungen	191
9.4.2	Zu viele Dimensionen	192
9.4.3	Linear nicht trennbare Datenpunktmengen	194

9.5	Ideen zum Weitermachen	195
9.6	Zusammenfassung und Ausblick	195
10	Neuronale Netze I: Das Häschenproblem	197
10.1	Bildererkennung: ein klassisches Problem	198
10.2	Was ist ein Modell?	199
10.3	Der Aufbau eines neuronalen Netzes	201
10.4	Das Häschenneuron und seine Kollegen	204
10.4.1	Die biologische Nervenzelle als Vorbild	205
10.4.2	Das künstliche Neuron	207
10.4.3	b steht für Bias	208
10.4.4	Die Aktivierungsfunktion	208
10.5	Das Beispielprogramm Tiere erkennen II	209
10.6	Der Code	211
10.7	Ideen zum Weitermachen	211
10.8	Zusammenfassung und Ausblick	212
11	Neuronale Netze II: Auf dem Weg ins Tal	213
11.1	Das überwachte Lernen	214
11.2	Die schrittweise Justierung des Modells	216
11.2.1	Die grundlegende Idee	217
11.2.2	Steigung	218
11.2.3	Tangente	219
11.2.4	Ableitung	219
11.2.5	Der Gradientenabstieg	220
11.2.6	Die Lernrate	222
11.3	Das Beispielprogramm Gradientenabstieg	223
11.4	Der Code	225
11.5	Tipps zum Weitermachen	226
11.6	Zusammenfassung und Ausblick	226

12	Neuronale Netze III: Fehler zurückverfolgen mit dem Neuronentrainer	229
<hr/>		
12.1	Was ist Backpropagation?	230
12.2	Das Beispielprogramm Neuronentrainer	231
12.2.1	Aufgaben und Netzarchitekturen	232
12.2.2	Ein Wiedersehen mit dem Häschenproblem	235
12.2.3	Lineare Trennbarkeit	236
12.3	Validierungsdaten, Überanpassung, Generatoren	237
12.3.1	Validierungsdaten und Überanpassung	238
12.3.2	Generatoren	239
12.4	Weitere Beispielaufgaben	240
12.4.1	Kreis und Hintergrund	240
12.4.2	Quadrat und Hintergrund	241
12.4.3	Farbtunnel sieben Farben	242
12.5	Die Anzahlen der verdeckten Schichten und der Neuronen	244
12.5.1	Viel hilft viel?	244
12.6	Was wir weggelassen haben	245
12.6.1	Dynamisierung der Lernrate	245
12.6.2	Batch und Epoche	245
12.6.3	Verlustfunktionen und Softmax	245
12.7	Ideen zum Weitermachen	246
12.8	Zusammenfassung und Ausblick	248
13	Neuronale Netze IV: Faltungsnetze, Autoencoder, GANs und DQL	249
<hr/>		
13.1	Faltungsnetze	249
13.1.1	Das Beispielprogramm Filterlabor I	250
13.1.2	Der Filterkernel in Aktion	251
13.1.3	Padding und Striding	253
13.1.4	Das Beispielprogramm Filterlabor II	254
13.1.5	Eine Filterkombination, die den Buchstaben K erkennt	255
13.1.6	Die Struktur eines Faltungsnetzes	256
13.1.7	Faltungsnetze trainieren	257

13.2	Modelle, die Bilder erzeugen	258
13.3	Autoencoder	260
13.3.1	Dimensionsreduktion	260
13.3.2	Daten ausdenken	261
13.4	Generative Adversarial Networks	261
13.5	Deep Q-Learning	264
13.5.1	Wie kommt die Umwelt ins Modell?	265
13.6	Zusammenfassung und Ausblick	265
14	Transformer verstehen	267
<hr/>		
14.1	Ein Sprachmodell von außen betrachtet	267
14.1.1	Unterschiede zum Markow-Prozess	268
14.2	Wörter in Zahlen codieren für Fortgeschrittene	269
14.3	Worteinbettungen	270
14.3.1	Relationen übertragen	274
14.4	Das Beispielprogramm Wort-Navigator	276
14.4.1	Textquelle I: Dünstende Köchinnen und frittierende Konditoren	276
14.4.2	Die Bedienoberfläche	277
14.4.3	Relationen übertragen im Wort-Navigator	279
14.5	Vom Text zur Worteinbettung	282
14.5.1	One Hot Encoding	283
14.5.2	Paare aus Ziel- und Kontextwörtern	284
14.5.3	Das Training	286
14.5.4	Vom Training zur Worteinbettung	287
14.5.5	Warum funktioniert das?	288
14.5.6	Kosinus-Ähnlichkeit	289
14.6	Vom Wort zum Satz zum Text: Sequenzeinbettungen	290
14.7	Das Beispielprogramm Sequenz-Navigator	292
14.7.1	Textquelle II: Ein Tag im Leben von Kati Katzenstein	292
14.7.2	Die Bedienoberfläche	294
14.7.3	Relationen zwischen Sequenzen	296

14.8	Transformer am Horizont	297
14.8.1	Ein verbesserter Markow-Prozess	297
14.8.2	Und täglich grüßt Frau Katzenstein	298
14.8.3	Positional Encodings	299
14.8.4	Gewichtungen	299
14.8.5	Sprache ist Kontext	299
14.8.6	Aufmerksamkeit	300
14.9	Zusammenfassung und Ausblick	301
14.9.1	Was wir ausgelassen haben	301
14.9.2	Unterschiede zu ausgewachsenen Transformatoren	302
14.9.3	Das erstaunliche Abstraktionsvermögen von Transformatoren	303
14.10	Ideen zum Weitermachen	304

Nachwort: Auf der Suche nach Trurls Elektrobarden	305
--	-----

Anhang	313
---------------	-----

A	Eine kurze Einführung in JavaScript und p5.js	315
B	Glossar	359
C	Quellen und weiterführende Literatur	369
D	Abbildungsverzeichnis	373

Index	377
-------------	-----

Materialien zum Buch

Auf der Webseite des Autors Pit Noack finden Sie alle Beispielprogramme zum direkten Ausprobieren im Browser und weitere Materialien zum Buch:

<https://www.maschinennah.de/ki-buch/>

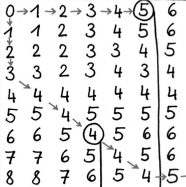
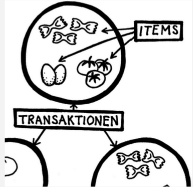
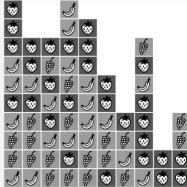
Kapitel 2 Texte bauen mit Markow	Kapitel 3 Schreibfehler automatisch korrigieren	Kapitel 4 Wörter gruppieren	Kapitel 5 Spiele für eine Person lösen
			
Beispielprogramm Nonsense-Texter	Beispielprogramm Wortvergleich	Beispielprogramm Assoziationsanalyse	Beispielprogramm Fruchtkräusch
STARTEN	STARTEN	STARTEN	STARTEN
<input type="checkbox"/> im Editor öffnen	<input type="checkbox"/> im Editor öffnen	<input type="checkbox"/> im Editor öffnen	<input type="checkbox"/> im Editor öffnen
Beispielprogramm Wörter vorschlagen	Beispielprogramm Korrekturvorschläge	> mehr...	> mehr...

Abbildung 1 Online können Sie die Projekte ausprobieren und den Code bearbeiten.

Auf der Verlagswebseite zu diesem Buch stehen Ihnen die Beispielprogramme 5 zusätzlich zum Download zur Verfügung.

Gehen Sie auf <http://www.rheinwerk-verlag.de/5810>. Klicken Sie auf den Reiter MATERIALIEN. Klicken Sie auf den Button HERUNTERLADEN, um den Download zu starten.

Künstliche Intelligenz verstehen

Hereinspaziert! Über ein Dutzend interaktive Projekte warten auf Ihre Neugierde. Schauen Sie zu, wie ein Algorithmus z. B. Gedichte schreibt oder Spiele gewinnt – oder gewinnen Sie?

Im Buch erfahren Sie, was jeweils dahintersteckt, und lernen einschlägige KI-Verfahren kennen. Von der Sprachkorrektur über Neuronale Netze bis zu Transformatoren, der Technologie hinter ChatGPT.

- + Grundlagen aus der Informatik
- + K-means-Clustering
- + Q-Learning
- + Transformer

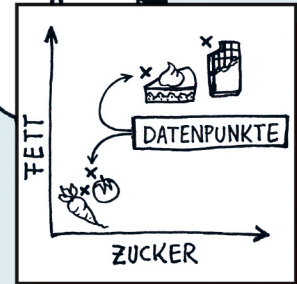
»Konsequent alltagsnah und ungewöhnlich unterhaltsam präsentiert dieses Lehrbuch wichtige KI-Themen.«



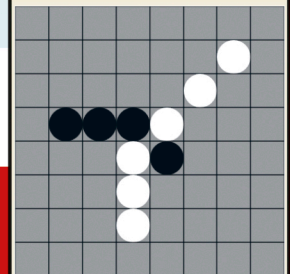
Alle Beispielprogramme online ausprobieren und mit dem Code experimentieren



Das Autorenteam verbindet Kunst mit KI, Sound mit Grafik und Spaß mit Code. Sie arbeiten digital, auf Leinwand, in transdisziplinären Projekten und in immer neuen Formaten. Pit Noack (Text & Code) begeistert in seinen Workshops für KI und Programmierung. Der Künstlerin Sophia Sanner (Illustration) ist kein Thema zu komplex, um es mit Cartoons und Infografiken auf humorvolle Weise anschaulich zu machen.

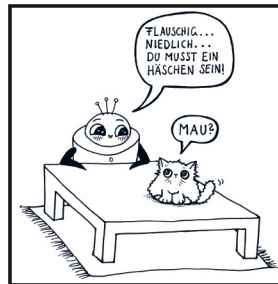


Reversi KI



Koordinaten anzeigen
 Alpha-Beta Pruning
Suchtiefe: 3

SCHWARZ (Mensch) IST AM ZUG
BEWERTUNG: 2
(Vorteil Maschine)



```
p5.js File Edit Sketch Help
Auto-refresh 07.01.q1erner by Maschin...
> q1ernerjs*
12 // Schnittstelle: wählt Aktion, führt s...
13 // und liefert eine modifizierte Umwe...
14" schritt(umwelt) {
15   const zustandsNr = umwelt.zustandsNr...
16   const aktionsNr = this.waehleAktion(...
17   const folgezustand = umwelt.aktion(...
18   const folgezustandsNr = folgezustand...
19   const belohnung = folgezustand.beloh...
20
21*
22 if (! umwelt.neueEpisode) {
23   this.aktualisiereQ(zustandsNr, akt...
24 }
25
26
27 return folgezustand;
28 }
```

