mitp

Maxim Lapan

Deep Reinforcement Learning

Das umfassende Praxis-Handbuch

Moderne Algorithmen für Chatbots, Robotik, diskrete Optimierung und Web-Automatisierung inkl. Multiagenten-Methoden

Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Der Verlag räumt Ihnen mit dem Kauf des ebooks das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und Einspeicherung und Verarbeitung in elektronischen Systemen.

Der Verlag schützt seine ebooks vor Missbrauch des Urheberrechts durch ein digitales Rechtemanagement. Bei Kauf im Webshop des Verlages werden die ebooks mit einem nicht sichtbaren digitalen Wasserzeichen individuell pro Nutzer signiert.

Bei Kauf in anderen ebook-Webshops erfolgt die Signatur durch die Shopbetreiber. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Neuerscheinungen, Praxistipps, Gratiskapitel,
Einblicke in den Verlagsalltag –
gibt es alles bei uns auf Instagram und Facebook

instagram.com/mitp_verlag

facebook.com/mitp.verlag

Maxim Lapan

Deep Reinforcement Learning

Das umfassende Praxis-Handbuch

Übersetzung aus dem Englischen
von Knut Lorenzen

mitp

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

ISBN 978-3-7475-0037-8
1. Auflage 2020

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Copyright ©Packt Publishing 2019
First published in the English language under the title 'Deep Reinforcement Learning Hands-On – Second Edition – (9781838826994)'

© 2020 mitp Verlags GmbH & Co. KG, Frechen

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Lektorat: Janina Bahlmann
Fachkorrektorat: Dr. Friedhelm Schwenker
Sprachkorrektorat: Petra Heubach-Erdmann
Coverbild: © agsandrew / stock.adobe.com
Satz: III-satz, Husby, www.drei-satz.de

Inhaltsverzeichnis

Über den Autor

Maxim Lapan ist Deep-Learning-Enthusiast und unabhängiger Forscher. Er verfügt über 15 Jahre Erfahrung als Softwareentwickler und Systemarchitekt. Er hat Linux-Kernel-Treiber entwickelt und verteilte Anwendungen entworfen und optimiert, die auf Tausenden Servern laufen. Er besitzt umfangreiche Erfahrung mit Big Data, Machine Learning und großen HPC- und Nicht-HPC-Systemen und hat das Talent, komplizierte Dinge in einfacher Sprache und mit anschaulichen Beispielen zu erklären. Derzeit beschäftigt er sich insbesondere mit praktischen Anwendungen des Deep Learnings, wie der Verarbeitung natürlicher Sprache (*Natural Language Processing*, NLP) und Deep Reinforcement Learning.

Maxim lebt mit seiner Familie in Moskau.

Ich möchte meiner Frau Olga und meinen Kindern Ksenia, Julia und Fedor für ihre Geduld und ihre Unterstützung danken. Dieses Buch zu schreiben stellte eine Herausforderung dar, und es wäre ohne euch nicht möglich gewesen, danke! Julia und Fedor haben beim Sammeln von Beispielen für MiniWoB (Kapitel 16, Navigation im Web) und beim Testen der Spielstärke des Vier-gewinnt-Agenten (Kapitel 23, AlphaZero Go) großartige Arbeit geleistet.

Über die Korrektoren

Mikhail Yurushkins Forschungsgebiete sind High-performance Computing und die Optimierung der Compiler-Entwicklung. Er ist Dozent an der SFEDU-Universität in Rostow am Don. Er gibt Kurse über fortgeschrittenes Deep Learning beim maschinellen Sehen und der Verarbeitung natürlicher Sprache. Er befasst sich seit mehr als acht Jahren mit plattformübergreifender Entwicklung in C++, Machine Learning und Deep Learning. Er ist Unternehmer und Gründer mehrerer Start-ups, unter anderem von BroutonLab, einem Data-Science-Unternehmen, das auf die Entwicklung KI-gestützter Softwareprodukte spezialisiert ist.

Per-Arne Andersen ist Doktorand an der Universität Agder in Norwegen und beschäftigt sich mit Reinforcement Learning. Er hat mehrere technische Arbeiten über den Einsatz von Reinforcement Learning bei Spielen verfasst und wurde für seine Forschung über modellbasiertes Reinforcement Learning von der British Computer Society als bester Student ausgezeichnet. Per-Arne Andersen ist außerdem Experte für Netzwerksicherheit und seit 2012 auf diesem Gebiet tätig. Seine aktuellen Forschungsinteressen sind Machine Learning, Deep Learning, Netzwerksicherheit und Reinforcement Learning.

Sergey Kolesnikov ist in Industrie und Wissenschaft als Forscher tätig und hat mehr als fünf Jahre Erfahrung mit Machine Learning, Deep Learning und Reinforcement Learning. Er arbeitet derzeit an industriellen Anwendungen, die Computervisualistik, Verarbeitung natürlicher Sprache und Empfehlungsdienste nutzen, und beteiligt sich an der Forschung zum Thema Reinforcement Learning. Er ist außerdem an Entscheidungsfindungsprozessen und Psychologie interessiert. Er ist Gewinner eines Wettbewerbs der Conference on Neural Information Processing Systems und Anhänger von Open Source. Darüber hinaus ist er Entwickler von Catalyst, einer High-Level-Umgebung für PyTorch zum Beschleunigen der Forschung und Entwicklung beim Deep Learning/Reinforcement Learning.

Über den Fachkorrektor der deutschen Ausgabe

Friedhelm Schwenker ist Privatdozent für Informatik (Fachgebiet: Machine Learning) an der Universität Ulm. Er hat im Bereich der Angewandten Mathematik promoviert und ist seit vielen Jahren im Bereich Machine Learning in Forschung und Lehre tätig. Seine Forschungsgebiete sind Pattern Recognition, Data Mining und Machine Learning mit Schwerpunkt Neuronale Netze. In jüngster Zeit befasst er sich auch mit Anwendungen des Machine Learnings im Affective Computing. Er ist Editor von 19 Proceedingsbänden und Special Issues sowie Autor von 200+ Journal- und Konferenzartikeln.

Einleitung

Das Thema dieses Buchs ist Reinforcement Learning (verstärkendes Lernen), ein Teilgebiet des Machine Learnings. Es konzentriert sich auf die anspruchsvolle Aufgabe, optimales Verhalten in komplexen Umgebungen zu erlernen. Der Lernvorgang wird ausschließlich durch den Wert einer Belohnung und durch Beobachtung der Umgebung gesteuert. Das Modell ist sehr allgemein und auf viele Situationen anwendbar, von einfachen Spielen bis hin zur Optimierung komplexer Fertigungsprozesse.

Aufgrund der Flexibilität und der allgemeinen Anwendbarkeit entwickelt sich das Fachgebiet Reinforcement Learning sehr schnell weiter und weckt großes Interesse bei Forschern, die vorhandene Methoden verbessern oder neue Methoden entwickeln wollen, und bei Praktikern, die ihre Aufgaben möglichst effizient bewältigen möchten.

Motivation

Dieses Buch stellt den Versuch dar, dem Mangel an praxisnahen und strukturierten Informationen über Verfahren und Ansätze des Reinforcement Learnings (RL) entgegenzuwirken. Es gibt weltweit umfassende Forschungsaktivitäten, und fast täglich werden neue Artikel veröffentlicht. Große Teile von Deep-Learning-Konferenzen wie NeurIPS (Neural Information Processing Systems) oder ICLR (International Conference on Learning Representations) widmen sich RL-Verfahren. Es gibt mehrere große Forschungsgruppen, die sich auf die Anwendung von RL-Verfahren in der Robotik, in der Medizin und auf Multiagenten-Systemen konzentrieren.

Die Informationen über die jüngsten Forschungsergebnisse sind zwar allgemein verfügbar, sie sind aber zu spezialisiert und zu abstrakt, um sie ohne erhebliche Anstrengung zu verstehen. Bei den praktischen Aspekten von RL-Anwendungen ist die Situation noch schlimmer, weil oft nicht klar ist, wie man von der abstrakten mathematischen Beschreibung einer Methode in einem Forschungsartikel zu einer funktionierenden Implementierung gelangt, die eine Aufgabe tatsächlich löst.

Das erschwert es am Fachgebiet Interessierten, die Methoden und Konzepte, die in Artikeln oder Konferenzvorträgen vorgestellt werden, unmittelbar zu verstehen. Es gibt ganz ausgezeichnete Blogbeiträge über verschiedene RL-Methoden, die durch funktionierende Beispiele veranschaulicht werden, aber das eingeschränkte Format eines Blogbeitrags ermöglicht es dem Autor nicht, mehr als ein oder zwei Methoden zu beschreiben, ohne den vollständigen Kontext darzustellen und zu zeigen, in welcher Beziehung die verschiedenen Methoden zueinanderstehen. Dieses Buch ist mein Versuch, dieses Problem in Angriff zu nehmen.

Der Ansatz

Ein weiterer Aspekt des Buchs ist die Praxisorientierung. Alle Methoden werden in verschiedenen Umgebungen implementiert, die von völlig trivial bis zu ziemlich komplex reichen. Ich habe versucht, die Beispiele so zu gestalten, dass sie leicht verständlich sind, was durch die Leistungsfähigkeit von PyTorch ermöglicht wurde. Die Komplexität und die Anforderungen der Beispiele orientieren sich an RL-Interessierten, die keinen Zugang zu sehr großer Rechenleistung haben, wie einem GPU-Cluster oder sehr leistungsstarken Workstations. Dadurch wird, wie ich hoffe, das hochinteressante und spannende Fachgebiet RL einem breiteren Publikum zugänglich, nicht nur Forschungsgruppen oder großen KI-Unternehmen. Allerdings geht es nach wie vor um **Deep** Reinforcement Learning, deshalb empfiehlt sich die Verwendung einer GPU. Etwa die Hälfte der Beispiele im Buch profitiert davon, wenn sie auf einer GPU ausgeführt werden.

Beim Reinforcement Learning kommen oft Umgebungen mittlerer Größe zum Einsatz, beispielsweise bei Atari-Spielen oder für kontinuierliche Steuerungsaufgaben, das Buch enthält aber auch einige Kapitel (Kapitel 10, 14, 15, 16 und 18), in denen größere Projekte beschrieben werden, um zu veranschaulichen, wie sich RL-Verfahren auf kompliziertere Umgebungen und Aufgaben anwenden lassen. Diese Beispiele sind allerdings auch keine vollständigen Projekte aus der Praxis (sonst würden sie ein eigenes Buch erfordern), sondern etwas umfassendere Aufgaben, die veranschaulichen, wie sich das RL-Paradigma auf Bereiche jenseits der üblichen Benchmarks anwenden lässt.

Bei den Beispielen in den ersten drei Teilen des Buchs habe ich versucht, den vollständigen Quellcode zu zeigen, damit die Beispiele eigenständig sind. In einigen Fällen führt das dazu, dass Teile des Codes wiederholt werden (beispielsweise sind die Trainings-Schleifen der meisten Verfahren sehr ähnlich), aber ich denke, direkt zu einer Methode von Interesse springen zu können, ist wichtiger, als einige Wiederholungen zu vermeiden. Alle Beispiele im Buch sind auf Github verfügbar (`https://github.com/PacktPublishing/Deep-Reinforcement-Learning-Hands-On-Second-Edition`) und Sie sind herzlich eingeladen, mit dem Code zu experimentieren und eigene Beiträge zu leisten.

Für wen ist das Buch gedacht?

Das Buch richtet sich vornehmlich an alle, die schon über einige Vorkenntnisse im Bereich Machine Learning verfügen und daran interessiert sind, Reinforcement Learning in der Praxis kennenzulernen. Der Leser sollte mit Python und den Grundlagen von Deep Learning und Machine Learning vertraut sein. Kenntnisse der Statistik und Wahrscheinlichkeitsrechnung sind von Vorteil, aber nicht unbedingt erforderlich, um den Großteil des Buchs zu verstehen.

Zum Inhalt des Buchs

Kapitel 1, Was ist Reinforcement Learning?, stellt eine Einführung in die grundlegenden Ideen des Reinforcement Learnings und der wichtigsten formalen Modelle dar.

Kapitel 2, OpenAI Gym, führt Sie anhand der Open-Source-Bibliothek Gym in die praxisnahen Aspekte des RL ein.

Kapitel 3, Deep Learning mit PyTorch, gibt einen Überblick über die PyTorch-Bibliothek.

Kapitel 4, Das Kreuzentropie-Verfahren, stellt eines der einfachsten RL-Verfahren vor, um Ihnen einen Eindruck von RL-Verfahren und RL-Aufgaben zu vermitteln.

Kapitel 5, Tabular Learning und das Bellman'sche Optimalitätsprinzip, führt in die wertebasierten RL-Verfahren ein.

Kapitel 6, Deep Q-Networks, beschreibt DQNs, die Erweiterung elementarer wertebasierter Verfahren, die es ermöglichen, Lösungen für komplexere Umgebungen zu finden.

Kapitel 7, Allgemeine RL-Bibliotheken, stellt die PTAN-Bibliothek vor, die wir im Buch verwenden werden, um die Implementierung von RL-Verfahren zu erleichtern.

Kapitel 8, DQN-Erweiterungen, gibt einen detaillierten Überblick über moderne Erweiterungen von DQNs, die zur Verbesserung der Stabilität und der Konvergenz in komplexen Umgebungen dienen.

Kapitel 9, Beschleunigung von RL-Verfahren, bietet eine Übersicht über die Möglichkeiten, die Ausführung von RL-Code zu beschleunigen.

Kapitel 10, Aktienhandel per Reinforcement Learning, erläutert das erste konkrete Projekt, die Anwendung des DQN-Verfahrens auf den Aktienhandel.

Kapitel 11, Eine Alternative: Policy Gradients, stellt eine weitere Familie wertebasierter RL-Verfahren vor, die auf Policy Learning beruhen.

Kapitel 12, Das Actor-Critic-Verfahren, beschreibt eines der am häufigsten verwendeten RL-Verfahren.

Kapitel 13, Asynchronous Advantage Actor Critic, erweitert das Actor-Critic-Verfahren durch parallele Kommunikation mit der Umgebung, um Stabilität und Konvergenz zu verbessern.

Kapitel 14, Chatbot-Training per Reinforcement Learning, beschreibt das zweite konkrete Projekt und zeigt, wie RL-Verfahren auf NLP-Aufgaben angewendet werden.

Kapitel 15, Die TextWorld-Umgebung, stellt die Anwendung von RL-Verfahren auf Spiele des Genres *Interactive Fiction* (IF) vor.

Kapitel 16, Navigation im Web, beschreibt ein weiteres größeres Projekt, nämlich die Anwendung von RL auf die Navigation im Web anhand von MiniWoB-Aufgaben.

Kapitel 17, Stetige Aktionsräume, erläutert die Eigenheiten von Umgebungen, die stetige Aktionsräume verwenden, und stellt weitere Verfahren vor.

Kapitel 18, RL in der Robotik, befasst sich mit der Anwendung von RL-Verfahren auf Aufgaben aus dem Gebiet der Robotik. In diesem Kapitel beschreibe ich die Entwicklung und das Training eines kleinen Hardware-Roboters mithilfe von RL-Verfahren.

Kapitel 19, Trust Regions – PPO, TRPO, ACKTR und SAC, ist ein weiteres Kapitel über stetige Aktionsräume und beschreibt die Trust-Region-Verfahren.

Kapitel 20, Blackbox-Optimierung beim Reinforcement Learning, beschreibt Optimierungsverfahren, die keine Gradienten in expliziter Form verwenden.

Kapitel 21, Fortgeschrittene Exploration, erörtert verschiedene Ansätze zur besseren Erkundung der Umgebung.

Kapitel 22, Jenseits modellfreier Verfahren – Imagination, stellt unter Berücksichtigung jüngster Forschungsergebnisse modellbasierte Ansätze für RL vor.

Kapitel 23, AlphaGo Zero, erläutert die Anwendung des AlphaGo-Zero-Verfahrens auf das Spiel »Vier gewinnt«.

Kapitel 24, RL und diskrete Optimierung, beschreibt die Anwendung von RL-Verfahren auf diskrete Optimierungen anhand einer Zauberwürfel-Umgebung (Rubik's Cube).

Kapitel 25, RL mit mehreren Agenten, stellt eine relative neue Entwicklungsrichtung bei RL-Verfahren für Situationen vor, in denen mehrere Agenten vorhanden sind.

Verwendung des Buchs

Alle Kapitel des Buchs, die RL-Verfahren beschreiben, sind identisch aufgebaut: Zunächst werden die Motivation für das Verfahren, die theoretischen Grundlagen und die zugrunde liegenden Ideen erläutert. Anschließend betrachten wir verschiedene Anwendungen des Verfahrens auf unterschiedliche Umgebungen anhand des vollständigen Quellcodes.

Sie können das Buch also auf verschiedene Weise verwenden:

1. Um sich einen schnellen Überblick über ein Verfahren zu verschaffen, können Sie den einführenden Teil des entsprechenden Kapitels lesen.
2. Um ein besseres Verständnis der Implementierung eines Verfahrens zu erlangen, können Sie sich den Quellcode ansehen und die dazugehörigen Kommentare lesen.
3. Um ein tiefer gehendes Verständnis eines Verfahrens zu erlangen, sollten Sie versuchen, es selbst zu implementieren und zum Laufen zu bringen (meiner Ansicht nach die beste Lernmethode). Dabei können Sie den vorhandenen Quellcode als Ausgangspunkt nutzen.

Ich kann mir nur wünschen, dass Ihnen das Buch von Nutzen sein wird!

Codebeispiele herunterladen

Die Codebeispiele aus diesem Buch können Sie unter `http://www.mitp.de/0036` herunterladen.

Farbige Abbildungen herunterladen

Eine farbige Version der Screenshots und Diagramme in diesem Buch finden Sie ebenfalls unter `http://www.mitp.de/0036` zum Download.

Konventionen im Buch

In diesem Buch werden verschiedene Textformatierungen verwendet, um zwischen Informationen unterschiedlicher Art zu unterscheiden. Nachstehend finden Sie einige Beispiele und deren Bedeutungen.

Schlüsselwörter, Datenbanktabellen-, Twitter-, Datei-, Ordner-, Datei- und Pfadnamen sowie URLs und Usereingaben werden im Fließtext in nicht proportionaler Schrift darge-

stellt. Zum Beispiel: »Mounten Sie die heruntergeladene Datei `WebStorm-10*.dmg` als weiteres Laufwerk Ihres Systems.«

Codeblöcke werden wie folgt dargestellt:

```
def grads_func(proc_name, net, device, train_queue):
    envs = [make_env() for _ in range(NUM_ENVS)]

    agent = ptan.agent.PolicyAgent(
        lambda x: net(x)[0], device=device,
        apply_softmax=True)

    exp_source = ptan.experience.ExperienceSourceFirstLast(
        envs, agent, gamma=GAMMA, steps_count=REWARD_STEPS)

    batch = []
    frame_idx = 0
    writer = SummaryWriter(comment=proc_name)
```

Eingaben oder Ausgaben auf der Kommandozeile sehen so aus:

```
rl_book_samples/Chapter11$ ./02_a3c_grad.py --cuda -n final
```

Neue Ausdrücke und **wichtige Begriffe** werden **fett** gedruckt. Auf dem Bildschirm auswählbare oder anklickbare Bezeichnungen, wie z. B. Menüpunkte oder Schaltflächen, werden in der Schriftart Kapitälchen gedruckt: »Nach einem Klick auf die Schaltfläche ABBRECHEN in der unteren rechten Ecke wird der Vorgang abgebrochen.«

Teil I

Grundlagen des Reinforcement Learnings

In diesem Teil:

Kapitel 1

Was ist Reinforcement Learning?

Reinforcement Learning (RL) ist ein Teilgebiet des Machine Learnings, das sich damit beschäftigt, zu lernen, mit der Zeit automatisch optimale Entscheidungen zu treffen. Dabei handelt es sich um ein allgemeines und gängiges Problem, das in vielen wissenschaftlichen und technischen Fachgebieten untersucht wird.

In unserer sich wandelnden Welt sind auch Aufgaben, die auf den ersten Blick wie statische Eingabe-Ausgabe-Aufgaben aussehen, aus übergeordneter Sicht dynamisch. Betrachten Sie beispielsweise die einfache überwachte Lernaufgabe, Bilder von Haustieren entweder als Hund oder als Katze zu klassifizieren. Sie haben die Trainingsdatenmenge zusammengestellt und den Klassifizierer mit dem Deep-Learning-Toolkit Ihrer Wahl implementiert, und nach einiger Zeit liefert das konvergierende Modell ausgezeichnete Ergebnisse. Gut so? Aber sicher! Sie haben das Modell zur Anwendung gebracht und lassen es weiterlaufen. Dann fahren Sie in den Urlaub, und nach Ihrer Rückkehr stellen Sie fest, dass modische Hundehaarschnitte inzwischen völlig anders aussehen und dass ein beträchtlicher Teil Ihrer Bilder fehlklassifiziert wird. Sie müssen also Ihre Trainingsbilder aktualisieren und den ganzen Vorgang wiederholen. Gut so? Natürlich nicht!

Dieses Beispiel soll zeigen, dass auch ganz einfache Aufgaben beim **Machine Learning** (**ML**) eine verborgene zeitliche Dimension besitzen, die häufig übersehen wird, bei einem Produktivsystem aber zu einem Problem werden kann. **Reinforcement Learning** (**RL**) ist ein Ansatz, der diese zusätzliche Dimension (für gewöhnlich die Zeit, das muss aber nicht so sein) in den Gleichungen der Lernregeln berücksichtigt und damit der menschlichen Vorstellung von einer künstlichen Intelligenz schon deutlich näher kommt.

In diesem Kapitel geht es um die folgenden Themen:

- Was RL mit anderen ML-Verfahren, nämlich überwachtem und unüberwachtem Lernen, gemeinsam hat und wodurch es sich von ihnen unterscheidet
- Die wichtigsten RL-Formalismen und in welcher Beziehung sie zueinander stehen
- Die theoretischen Grundlagen des RL: der Markov-Entscheidungsprozess

1.1 Überwachtes Lernen

Überwachtes Lernen dürfte Ihnen bekannt sein, denn es ist die am häufigsten untersuchte und am besten bekannte Machine-Learning-Aufgabe. Die grundlegende Frage lautet: Wie kann man automatisch eine Funktion finden, die einer Eingabe anhand einer Menge von Beispielpaaren eine bestimmte Ausgabe zuordnet? So formuliert klingt die Aufgabe einfach, aber sie ist mit vielen kniffligen Problemen verbunden, die Computer erst in jüngster Zeit einigermaßen erfolgreich lösen konnten. Es gibt eine Vielzahl von Beispielen für überwachte Lernaufgaben, wie beispielsweise diese:

- **Textklassifikation:** Handelt es sich bei einer E-Mail um Spam oder nicht?
- **Bildklassifikation und Objektlokalisierung:** Zeigt ein Bild eine Katze, einen Hund oder etwas anderes?
- **Regressionen:** Wie wird den Messwerten der Wetterstation zufolge das morgige Wetter?
- **Stimmungsanalyse:** Wie zufrieden ist ein Kunde, der eine Rezension geschrieben hat?

Die Fragestellungen können sich unterscheiden, ihnen liegt jedoch die gleiche Idee zugrunde: Es liegen viele Beispiele für Eingaben und der dazugehörigen Ausgaben vor und wir möchten erlernen, die Ausgabe für neue, noch unbekannte Eingaben zu erzeugen. Die Bezeichnung *überwachtes* Lernen ist der Tatsache geschuldet, dass das System anhand bekannter Antworten lernt, denn die Beispiele sind korrekt mit Labels gekennzeichnet.

1.2 Unüberwachtes Lernen

Auf der anderen Seite gibt es das unüberwachte Lernen, bei dem die Daten nicht mit einem Label versehen sind. Das Ziel ist es, in den vorliegenden Daten eine verborgene Struktur aufzuspüren. Ein gängiges Beispiel für diesen Lernansatz ist das Clustering von Daten. Der Algorithmus versucht dabei, die Datenobjekte in Cluster aufzuteilen und auf diese Weise Beziehungen in den Daten aufzudecken. Man könnte beispielsweise nach ähnlichen Bildern suchen oder nach Kunden, die sich ähnlich verhalten.

Eine weitere unüberwachte Lernmethode, die sich zunehmender Beliebtheit erfreut, sind **Generative Adversarial Networks (GANs)**. Dabei gibt es zwei konkurrierende neuronale Netze. Das erste erzeugt gefälschte Daten, und das zweite versucht, zu unterscheiden, ob die Daten gefälscht wurden oder zur echten Datenmenge gehören. Im Lauf der Zeit können die beiden Netze ihre jeweilige Aufgabe immer besser erfüllen, indem sie subtile Muster in der Datenmenge erfassen.

1.3 Reinforcement Learning

RL ist irgendwo zwischen vollständig überwachtem Lernen und dem vollständigen Fehlen vordefinierter Labels einzuordnen. Einerseits verwendet es viele wohlbekannte Methoden des überwachten Lernens, wie tiefe neuronale Netze zur Funktionsapproximation, stochastischem Gradientenabstieg und Backpropagation, um Datenrepräsentationen zu erlernen. Andererseits werden diese Methoden dabei für gewöhnlich auf andere Art und Weise angewendet.

In den nächsten beiden Abschnitten dieses Kapitels haben wir die Gelegenheit, bestimmte Details des RL-Ansatzes zu erkunden, wie die Annahmen und Abstraktionen in mathematisch strenger Form. Fürs Erste verwenden wir eine weniger formale und anschaulichere Beschreibung, um RL mit überwachtem und unüberwachtem Lernen zu vergleichen.

Nehmen wir an, es gibt einen Agenten, der in einer Umgebung bestimmte Aktionen ausführen soll. Eine Robotermaus in einem Labyrinth ist ein gutes Beispiel, wir könnten uns aber auch einen automatischen Helikopter vorstellen, der versucht, ein Flugmanöver zu vollführen, oder ein Schachprogramm, das lernt, wie man einen Großmeister schlägt. Der Einfachheit halber bleiben wir bei der Robotermaus.

Ihre Umgebung ist ein Labyrinth, in dem es an einigen Stellen Futter gibt und an anderen Stromschläge. Die Robotermaus kann verschiedene Aktionen ausführen, wie etwa sich

nach links oder rechts zu drehen oder sich vorwärts zu bewegen. Sie kann zu jedem Zeitpunkt den vollständigen Zustand des Labyrinths beobachten, um zu entscheiden, welche Aktion sie ausführt (Abbildung 1.1). Sie versucht, so viel Futter wie möglich zu finden und Stromschläge möglichst zu vermeiden. Die Signale Futter und Stromschlag sind die *Belohnung*, die der Agent (die Robotermaus) von der Umgebung als zusätzliches Feedback zu den Aktionen erhält. Belohnung ist beim RL ein sehr wichtiges Konzept, auf das ich später in diesem Kapitel noch kommen werde. An dieser Stelle genügt es, zu wissen, dass es das Ziel des Agenten ist, eine möglichst große kumulierte Belohnung zu erhalten. In diesem speziellen Beispiel könnte die Maus einen Stromschlag in Kauf nehmen, um an eine Stelle zu gelangen, an der es viel Futter gibt. Das wäre ein besseres Ergebnis, als einfach nur stehen zu bleiben und gar kein Futter zu finden.

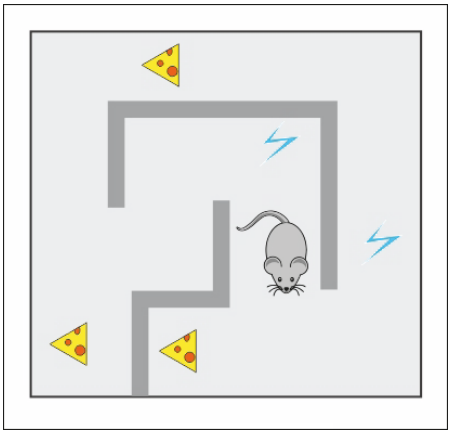

Abb. 1.1: Das Labyrinth der Robotermaus

Wir wollen das Wissen über die Umgebung und die jeweils beste Aktion in einer bestimmten Situation in der Robotermaus nicht fest einprogrammieren – das wäre zu aufwendig und würde nutzlos werden, wenn sich das Labyrinth auch nur leicht ändert. Wir benötigen vielmehr ein paar Methoden, die es dem Roboter ermöglichen, selbst zu erlernen, Stromschläge zu vermeiden und so viel Futter wie möglich zu finden.

Reinforcement Learning bietet hier eine Lösung, die sich von überwachten und unüberwachten Lernmethoden unterscheidet. Es gibt keine vordefinierten Labels wie beim überwachten Lernen. Niemand kennzeichnet die Bilder, die der Roboter sieht, mit *gut* oder *schlecht* oder gibt ihm Hinweise, in welche Richtung er sich am besten bewegen sollte.

Wir sind allerdings auch nicht völlig blind wie beim unüberwachten Lernen – es gibt ein Belohnungssystem. Belohnungen können positiv (beim Finden von Futter), negativ (bei einem Stromschlag) oder neutral sein (wenn nichts Besonderes geschieht). Indem er die Belohnungen beobachtet und in Beziehung zur ausgeführten Aktion setzt, kann unser Agent erlernen, wie er eine Aktion besser ausführen kann, mehr Futter findet und weniger Stromschläge erhält.

Natürlich hat die allgemeine Anwendbarkeit und die Flexibilität des Reinforcement Learnings ihren Preis. RL gilt als sehr viel schwieriger als überwachtes und unüberwachtes Lernen. Werfen wir einen kurzen Blick darauf, was Reinforcement Learning so knifflig macht.

1.4 Herausforderungen beim Reinforcement Learning

Zunächst einmal ist zu beachten, dass die Beobachtung beim RL vom Verhalten des Agenten abhängt und in gewissem Maße sogar das *Ergebnis* seines Verhaltens ist. Wenn Ihr Agent sich ineffizient verhält, sagt die Beobachtung nichts darüber aus, was er falsch gemacht hat und was man unternehmen sollte, um das Ergebnis zu verbessern (der Agent erhält die ganze Zeit nur negatives Feedback). Wenn der Agent stur ist und weiterhin Fehler begeht, kann die Beobachtung den falschen Eindruck vermitteln, dass es keine Möglichkeit gibt, eine größere Belohnung zu erhalten – das Leben ist ein Leidensweg –, was aber völlig falsch sein könnte.

Im Machine-Learning-Jargon spricht man davon, dass die **i.i.d.-Annahme**. nicht erfüllt ist. Diese Annahme besagt, dass die Daten **unabhängig** voneinander (*independent*) und **identisch verteilt** (*identically distributed*) sind, was für die meisten überwachten Lernmethoden erforderlich ist.

Darüber hinaus wird das Leben unseres Agenten dadurch verkompliziert, dass er nicht nur die erlernte Policy **nutzen** (engl. *exploit*), sondern die Umgebung aktiv **erkunden** (engl. *explore*) muss, denn möglicherweise könnten wir ein erheblich besseres Ergebnis erzielen, wenn wir anders vorgehen. Das Problem ist nur, dass eine zu ausführliche Exploration zu einer beträchtlichen Verringerung der Belohnung führen könnte (ganz zu schweigen davon, dass der Agent auch *vergessen* kann, was er zuvor erlernt hat). Wir müssen also irgendwie ein Gleichgewicht zwischen diesen beiden Vorgehensweisen finden. Wie sich ein Ausweg aus diesem Dilemma zwischen Exploitation und Exploration finden lässt, ist eine der grundlegenden offenen Fragen beim RL.

Menschen müssen solche Entscheidungen ständig treffen: Soll ich zum Abendessen einen bekannten Ort aufsuchen oder doch das schicke neue Restaurant ausprobieren? Wie häufig sollte man den Arbeitsplatz wechseln? Sollte man sich mit einem neuen Fachgebiet befassen oder auf dem jetzigen weiterarbeiten? Eine allgemeingültige Antwort auf diese Fragen gibt es nicht.

Der dritte Faktor, der die Sache verkompliziert, ist die Tatsache, dass die Belohnung einer Aktion unter Umständen mit erheblicher Verzögerung erfolgt. Beim Schach kann ein einziger starker Zug in der Mitte der Partie spielentscheidend sein. Beim Lernen müssen wir so etwas erkennen, was schwierig sein kann, wenn wir mit dem Spielverlauf und den Aktionen beschäftigt sind.

Trotz all dieser Hindernisse und Komplikationen hat RL in den letzten Jahren große Fortschritte erzielt und wird immer mehr zu einem Fachgebiet für Forschung und praktische Anwendungen.

Neugierig geworden? Sehen wir uns die Details an und betrachten die RL-Formalismen und die geltenden Spielregeln.

1.5 RL-Formalismen

Bei jedem wissenschaftlichen Fachgebiet gibt es bestimmte Annahmen und Einschränkungen. Im letzten Abschnitt habe ich überwachtes Lernen erläutert, bei dem die Kenntnis der Eingabe-Ausgabe-Paare vorausgesetzt wird. Es gibt für die Daten keine Labels? Nichts für ungut, Sie müssen die Labels irgendwie beschaffen oder einen anderen Ansatz verfolgen. Überwachtes Lernen ist deswegen nicht besser oder schlechter als andere Verfahren, es ist