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)

Inhaltsverzeichnis

Impressum

Über den Autor

Über die Korrektoren

Über den Fachkorrektor der deutschen Ausgabe

Einleitung

Teil I: Grundlagen des Reinforcement Learnings

Kapitel 1: Was ist Reinforcement Learning?

- 1.1 Überwachtes Lernen
- 1.2 Unüberwachtes Lernen
- 1.3 Reinforcement Learning
- 1.4 Herausforderungen beim Reinforcement Learning
- 1.5 RL-Formalismen
  - 1.5.1 Belohnung
  - 1.5.2 Der Agent
  - 1.5.3 Die Umgebung
  - 1.5.4 Aktionen
  - 1.5.5 Beobachtungen
- 1.6 Die theoretischen Grundlagen des Reinforcement Learnings
  - 1.6.1 Markov-Entscheidungsprozesse
  - 1.6.2 Markov-Prozess

1.6.3 Markov-Belohnungsprozess
1.6.4 Aktionen hinzufügen
1.6.5 Policy
1.7 Zusammenfassung

Kapitel 2: OpenAI Gym

2.1 Aufbau des Agenten
2.2 Anforderungen an Hard- und Software
2.3 OpenAI-Gym-API
2.3.1 Aktionsraum
2.3.2 Beobachtungsraum
2.3.3 Die Umgebung
2.3.4 Erzeugen der Umgebung
2.3.5 Die CartPole-Sitzung
2.4 Ein CartPole-Agent nach dem Zufallsprinzip
2.5 Zusätzliche Gym-Funktionalität: Wrapper und Monitor
2.5.1 Wrapper
2.5.2 Monitor
2.6 Zusammenfassung

Kapitel 3: Deep Learning mit PyTorch

3.1 Tensoren
3.1.1 Tensoren erzeugen
3.1.2 Skalare Tensoren
3.1.3 Tensor-Operationen
3.1.4 GPU-Tensoren
3.2 Gradienten
3.2.1 Tensoren und Gradienten

3.3 NN-Bausteine
3.4 Benutzerdefinierte Schichten
3.5 Verlustfunktionen und Optimierer
3.5.1 Verlustfunktionen
3.5.2 Optimierer
3.6 Monitoring mit TensorBoard
3.6.1 Einführung in TensorBoard
3.6.2 Plotten
3.7 Beispiel: GAN für Bilder von Atari-Spielen
3.8 PyTorch Ignite
3.8.1 Konzepte
3.9 Zusammenfassung

Kapitel 4: Das Kreuzentropie-Verfahren

4.1 Klassifikation von RL-Verfahren
4.2 Kreuzentropie in der Praxis
4.3 Kreuzentropie beim CartPole
4.4 Kreuzentropie beim FrozenLake
4.5 Theoretische Grundlagen des Kreuzentropie-Verfahrens
4.6 Zusammenfassung

Teil II: Wertebasierte Verfahren

Kapitel 5: Tabular Learning und das Bellman’sche Optimalitätsprinzip

5.1 Wert, Zustand und Optimalität
5.2 Das Bellman’sche Optimalitätsprinzip

5.3 Aktionswert
5.4 Wertiteration
5.5 Wertiteration in der Praxis
5.6 Q-Learning in der FrozenLake-Umgebung
5.7 Zusammenfassung

Kapitel 6: Deep Q-Networks

6.1 Wertiteration in der Praxis
6.2 Tabular Q-Learning
6.3 Deep Q-Learning
6.3.1 Interaktion mit der Umgebung
6.3.2 SGD-Optimierung
6.3.3 Korrelation der Schritte
6.3.4 Die Markov-Eigenschaft
6.3.5 Die endgültige Form des DQN-Trainings
6.4 DQN mit Pong
6.4.1 Wrapper
6.4.2 DQN-Modell
6.4.3 Training
6.4.4 Ausführung und Leistung
6.4.5 Das Modell in Aktion
6.5 Weitere Möglichkeiten
6.6 Zusammenfassung

Kapitel 7: Allgemeine RL-Bibliotheken

7.1 Warum RL-Bibliotheken?
7.2 Die PTAN-Bibliothek
7.2.1 Aktionsselektoren

7.2.2 Der Agent
7.2.3 Quelle der Erfahrungswerte
7.2.4 Replay Buffer für Erfahrungswerte
7.2.5 Die TargetNet-Klasse
7.2.6 Hilfsfunktionen für Ignite
7.3 Lösung der CartPole-Umgebung mit PTAN
7.4 Weitere RL-Bibliotheken
7.5 Zusammenfassung

Kapitel 8: DQN-Erweiterungen

8.1 Einfaches DQN
8.1.1 Die Bibliothek common
8.1.2 Implementierung
8.1.3 Ergebnisse
8.2 N-Schritt-DQN
8.2.1 Implementierung
8.2.2 Ergebnisse
8.3 Double DQN
8.3.1 Implementierung
8.3.2 Ergebnisse
8.4 Verrauschte Netze
8.4.1 Implementierung
8.4.2 Ergebnisse
8.5 Priorisierter Replay Buffer
8.5.1 Implementierung
8.5.2 Ergebnisse
8.6 Rivalisierendes DQN
8.6.1 Implementierung

8.6.2 Ergebnisse
8.7 Kategoriales DQN
8.7.1 Implementierung
8.7.2 Ergebnisse
8.8 Alles miteinander kombinieren
8.8.1 Ergebnisse
8.9 Zusammenfassung
8.10 Quellenangaben

Kapitel 9: Beschleunigung von RL-Verfahren

9.1 Die Bedeutung der Geschwindigkeit
9.2 Der Ausgangspunkt
9.3 Der Berechnungsgraph in PyTorch
9.4 Mehrere Umgebungen
9.5 Spielen und Trainieren in separaten Prozessen
9.6 Optimierung der Wrapper
9.7 Zusammenfassung der Benchmarks
9.8 Atari-Emulation: CuLE
9.9 Zusammenfassung
9.10 Quellenangaben

Kapitel 10: Aktienhandel per Reinforcement Learning

10.1 Börsenhandel
10.2 Daten
10.3 Aufgabenstellungen und Grundsatzentscheidungen
10.4 Die Handelsumgebung
10.5 Modelle

10.6 Trainingscode
10.7 Ergebnisse
10.7.1 Das Feedforward-Modell
10.7.2 Das Faltungsmodell
10.8 Weitere Möglichkeiten
10.9 Zusammenfassung

Teil III: Policybasierte Verfahren

Kapitel 11: Eine Alternative: Policy Gradients
11.1 Werte und Policy
11.1.1 Warum Policy?
11.1.2 Repräsentation der Policy
11.1.3 Policy Gradients
11.2 Das REINFORCE-Verfahren
11.2.1 Das CartPole-Beispiel
11.2.2 Ergebnisse
11.2.3 Policybasierte und wertebasierte Verfahren
11.3 Probleme mit REINFORCE
11.3.1 Notwendigkeit vollständiger Episoden
11.3.2 Große Varianz der Gradienten
11.3.3 Exploration
11.3.4 Korrelation zwischen Beispielen
11.4 PG mit CartPole
11.4.1 Implementierung
11.4.2 Ergebnisse
11.5 PG mit Pong
11.5.1 Implementierung
11.5.2 Ergebnisse

11.6 Zusammenfassung

Kapitel 12: Das Actor-Critic-Verfahren

12.1 Verringern der Varianz

12.2 Varianz der CartPole-Umgebung

12.3 Actor-Critic

12.4 A2C mit Pong

12.5 A2C mit Pong: Ergebnisse

12.6 Optimierung der Hyperparameter

12.6.1 Lernrate

12.6.2 Beta

12.6.3 Anzahl der Umgebungen

12.6.4 Batchgröße

12.7 Zusammenfassung

Kapitel 13: Asynchronous Advantage Actor Critic

13.1 Korrelation und Stichprobeneffizienz

13.2 Ein weiteres A zu A2C hinzufügen

13.3 Multiprocessing in Python

13.4 A3C mit Datenparallelität

13.4.1 Implementierung

13.4.2 Ergebnisse

13.5 A3C mit Gradientenparallelität

13.5.1 Implementierung

13.5.2 Ergebnisse

13.6 Zusammenfassung

Kapitel 14: Chatbot-Training per Reinforcement Learning

- 14.1 Chatbots – ein Überblick
- 14.2 Chatbot-Training
- 14.3 Grundlagen der Verarbeitung natürlicher Sprache
  - 14.3.1 Rekurrente neuronale Netze
  - 14.3.2 Wort-Embeddings
  - 14.3.3 Encoder-Decoder
- 14.4 Seq2Seq-Training
  - 14.4.1 Log-Likelihood-Training
  - 14.4.2 Der BLEU-Score
  - 14.4.3 RL und Seq2Seq
  - 14.4.4 Self-critical Sequence Training
- 14.5 Das Chatbot-Beispiel
  - 14.5.1 Aufbau des Beispiels
  - 14.5.2 Module: cornell.py und data.py
  - 14.5.3 BLEU-Score und utils.py
  - 14.5.4 Modell
- 14.6 Daten überprüfen
- 14.7 Training: Kreuzentropie
  - 14.7.1 Implementierung
  - 14.7.2 Ergebnisse
- 14.8 Training: Self-critical Sequence Training (SCST)
  - 14.8.1 Implementierung
  - 14.8.2 Ergebnisse
- 14.9 Tests der Modelle mit Daten
- 14.10 Telegram-Bot
- 14.11 Zusammenfassung

Kapitel 15: Die TextWorld-Umgebung

15.1 Interactive Fiction

15.2 Die Umgebung

15.2.1 Installation

15.2.2 Spiel erzeugen

15.2.3 Beobachtungs- und Aktionsräume

15.2.4 Zusätzliche Informationen

15.3 Einfaches DQN

15.3.1 Vorverarbeitung von Beobachtungen

15.3.2 Embeddings und Encoder

15.3.3 DQN-Modell und Agent

15.3.4 Trainingscode

15.3.5 Trainingsergebnisse

15.4 Das Modell für den Befehlsgenerator

15.4.1 Implementierung

15.4.2 Ergebnisse des Pretrainings

15.4.3 DQN-Trainingscode

15.4.4 Ergebnis des DQN-Trainings

15.5 Zusammenfassung

Kapitel 16: Navigation im Web

16.1 Webnavigation

16.1.1 Browserautomatisierung und RL

16.1.2 Mini World of Bits

16.2 OpenAI Universe

16.2.1 Installation

16.2.2 Aktionen und Beobachtungen

16.2.3 Umgebung erzeugen

16.2.4 MiniWoB-Stabilität

16.3 Einfaches Anklicken
16.3.1 Aktionen auf dem Gitter
16.3.2 Übersicht der Beispiele
16.3.3 Modell
16.3.4 Trainingscode
16.3.5 Container starten
16.3.6 Trainingsprozess
16.3.7 Überprüfen der erlernten Policy
16.3.8 Probleme mit einfachem Anklicken
16.4 Demonstrationen durch den Menschen
16.4.1 Aufzeichnung von Demonstrationen
16.4.2 Aufzeichnungsformat
16.4.3 Training durch Demonstration
16.4.4 Ergebnisse
16.4.5 Tic-Tac-Toe
16.5 Hinzufügen von Beschreibungstext
16.5.1 Implementierung
16.5.2 Ergebnisse
16.6 Weitere Möglichkeiten
16.7 Zusammenfassung

Teil IV: Fortgeschrittene Verfahren und Techniken

Kapitel 17: Stetige Aktionsräume

17.1 Wozu stetige Aktionsräume?
17.2 Aktionsraum
17.3 Umgebungen
17.4 Das A2C-Verfahren
17.4.1 Implementierung

17.4.2 Ergebnisse
17.4.3 Modelle verwenden und Videos aufzeichnen
17.5 Deterministisches Policy-Gradienten-Verfahren
17.5.1 Exploration
17.5.2 Implementierung
17.5.3 Ergebnisse
17.5.4 Videos aufzeichnen
17.6 Distributional Policy Gradients
17.6.1 Architektur
17.6.2 Implementierung
17.6.3 Ergebnisse
17.6.4 Videoaufzeichnung
17.7 Weitere Möglichkeiten
17.8 Zusammenfassung

Kapitel 18: RL in der Robotik

18.1 Roboter und Robotik
18.1.1 Komplexität von Robotern
18.1.2 Hardware
18.1.3 Plattform
18.1.4 Sensoren
18.1.5 Aktuatoren
18.1.6 Rahmen
18.2 Ein erstes Trainingsziel
18.3 Emulator und Modell
18.3.1 Definitionsdatei des Modells
18.3.2 Die robot-Klasse
18.4 DDPG-Training und Ergebnisse

18.5 Steuerung der Hardware
18.5.1 MicroPython
18.5.2 Handhabung von Sensoren
18.5.3 Servos ansteuern
18.5.4 Einrichtung des Modells auf der Hardware
18.5.5 Alles kombinieren
18.6 Experimente mit der Policy
18.7 Zusammenfassung

Kapitel 19: Trust Regions – PPO, TRPO, ACKTR und SAC

19.1 Roboschool
19.2 Standard-A2C-Verfahren
19.2.1 Implementierung
19.2.2 Ergebnisse
19.2.3 Videoaufzeichnungen
19.3 Proximal Policy Optimization (PPO)
19.3.1 Implementierung
19.3.2 Ergebnisse
19.4 Trust Region Policy Optimization (TRPO)
19.4.1 Implementierung
19.4.2 Ergebnisse
19.5 Advantage Actor-Critic mit Kronecker-Factored Trust Region (ACKTR)
19.5.1 Implementierung
19.5.2 Ergebnisse
19.6 Soft-Actor-Critic (SAC)
19.6.1 Implementierung
19.6.2 Ergebnisse

19.7 Zusammenfassung

Kapitel 20: Blackbox-Optimierung beim Reinforcement Learning

20.1 Blackbox-Verfahren

20.2 Evolutionsstrategien (ES)

20.3 ES mit CartPole

20.3.1 Ergebnisse

20.4 ES mit HalfCheetah

20.4.1 Implementierung

20.4.2 Ergebnisse

20.5 Genetische Algorithmen (GA)

20.6 GA mit CartPole

20.6.1 Ergebnisse

20.7 GA-Optimierung

20.7.1 Deep GA

20.7.2 Novelty Search

20.8 GA mit HalfCheetah

20.8.1 Ergebnisse

20.9 Zusammenfassung

20.10 Quellenangaben

Kapitel 21: Fortgeschrittene Exploration

21.1 Die Bedeutung der Exploration

21.2 Was ist das Problem beim ϵ-Greedy-Ansatz?

21.3 Alternative Explorationsverfahren

21.3.1 Verrauschte Netze

21.3.2 Zählerbasierte Verfahren

21.3.3 Vorhersagebasierte Verfahren
21.4 MountainCar-Experimente
21.4.1 Das DQN-Verfahren mit ϵ-Greedy-Ansatz
21.4.2 Das DQN-Verfahren mit verrauschten Netzen
21.4.3 Das DQN-Verfahren mit Zustandszählern
21.4.4 Das PPO-Verfahren
21.4.5 Das PPO-Verfahren mit verrauschten Netzen
21.4.6 Das PPO-Verfahren mit zählerbasierter Exploration
21.4.7 Das PPO-Verfahren mit Netz-Destillation
21.5 Atari-Experimente
21.5.1 Das DQN-Verfahren mit ϵ-Greedy-Ansatz
21.5.2 Das klassische PPO-Verfahren
21.5.3 Das PPO-Verfahren mit Netz-Destillation
21.5.4 Das PPO-Verfahren mit verrauschten Netzen
21.6 Zusammenfassung
21.7 Quellenangaben

Kapitel 22: Jenseits modellfreier Verfahren – Imagination

22.1 Modellbasierte Verfahren
22.1.1 Modellbasierte und modellfreie Verfahren
22.2 Unzulänglichkeiten der Modelle
22.3 Imagination-augmented Agent
22.3.1 Das Umgebungsmodell
22.3.2 Die Rollout-Policy
22.3.3 Der Rollout-Encoder
22.3.4 Ergebnisse der Arbeit
22.4 I2A mit dem Atari-Spiel Breakout

22.4.1 Der Standard-A2C-Agent
22.4.2 Training des Umgebungsmodells
22.4.3 Der Imagination-Agent
22.5 Ergebnisse der Experimente
22.5.1 Der Basis-Agent
22.5.2 Training der EM-Gewichte
22.5.3 Training mit dem I2A-Modell
22.6 Zusammenfassung
22.7 Quellenangaben

Kapitel 23: AlphaGo Zero

23.1 Brettspiele
23.2 Das AlphaGo-Zero-Verfahren
23.2.1 Überblick
23.2.2 Monte-Carlo-Baumsuche
23.2.3 Self-Playing
23.2.4 Training und Bewertung
23.3 Vier-gewinnt-Bot
23.3.1 Spielmodell
23.3.2 Implementierung der Monte-Carlo-Baumsuche
23.3.3 Modell
23.3.4 Training
23.3.5 Test und Vergleich
23.4 Vier gewinnt: Ergebnisse
23.5 Zusammenfassung
23.6 Quellenangaben

Kapitel 24: RL und diskrete Optimierung

24.1 Die Reputation von Reinforcement Learnings
24.2 Zauberwürfel und kombinatorische Optimierung
24.3 Optimalität und Gottes Zahl
24.4 Ansätze zur Lösung
24.4.1 Datenrepräsentation
24.4.2 Aktionen
24.4.3 Zustände
24.5 Trainingsvorgang
24.5.1 Architektur des neuronalen Netzes
24.5.2 Training
24.6 Anwendung des Modells
24.7 Ergebnisse der Arbeit
24.8 Code
24.8.1 Würfel-Umgebungen
24.8.2 Training
24.8.3 Suchvorgang
24.9 Ergebnisse des Experiments
24.9.1 Der 2x2-Würfel
24.9.2 Der 3x3-Würfel
24.9.3 Weitere Verbesserungen und Experimente
24.10 Zusammenfassung

Kapitel 25: RL mit mehreren Agenten

25.1 Mehrere Agenten
25.1.1 Kommunikationsformen
25.1.2 Der RL-Ansatz
25.2 Die MAgent-Umgebung
25.2.1 Installation

25.2.2 Überblick
25.2.3 Eine zufällige Umgebung
25.3 Deep Q-Networks für Tiger
25.3.1 Training und Ergebnisse
25.4 Zusammenarbeit der Tiger
25.5 Training der Tiger und Hirsche
25.6 Der Kampf ebenbürtiger Akteure
25.7 Zusammenfassung

Maxim Lapan

Deep Reinforcement Learning

Das umfassende Praxis-Handbuch

Übersetzung aus dem Englischen
von Knut Lorenzen

Impressum

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 978-3-7475-0038-5
1. Auflage 2020

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

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

Dieses Ebook verwendet das ePub-Format und ist optimiert für die Nutzung mit dem iBooks-reader auf dem iPad von Apple. Bei der Verwendung anderer Reader kann es zu Darstellungsproblemen kommen.

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

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-