Technische Hochschule Ostwestfalen-Lippe
University of Applied Sciences and Arts

Wintersemester 2022/2023

Vergleich verschiedener Reinforcement Learning Algorithmen

Modularbeit

Vorgelegt im Kontext des Moduls „Anwendung des maschinellen Lernens“

am Fachbereich Technische Informatik und Elektrotechnik
im Studiengang Data Science

Veranstalter:	Prof. Dr. Philipp Bruland
Vorgelegt von:	Bjarne Seen
	Liebigstraße 130
	32657 Lemgo
	bjarne.seen@stud.th-owl.de
Matr. Nr.:	15467085
Vorgelegt von:	Joshua Henjes
	Hanseweg 11
	32657 Lemgo
	joshua.henjes@stud.th-owl.de
Matr. Nr.:	15467024
Abgabetermin:	02.03.2023

27. Februar 2023

Inhaltsverzeichnis

Abkürzungen

API	Application Programming Interface

1 Einleitung

Reinforcement Learning ist neben Supervised und Unsupervised Learning eins der elementaren Felder des maschinellen Lernens. Im Gegensatz zu den anderen Feldern benötigt Reinforcement Learning keine Trainingsdaten, denn der Algorithmus lernt durch wiederholtes Interagieren mit einer dynamischen Umgebung eine Strategie, um eine Belohnungsmetrik zu maximieren. Es wird daher auch als bestärktes lernen oder verstärktes lernen bezeichnet.

Bekannt wurde das Reinforcement Learning vor allem durch das Meistern von bekannten Brett- und Computerspielen, so ist Googles „AlphaGo“ in der Lage, die besten Go Spieler der Welt zu schlagen. Trotz dieser beeindruckenden Erfolge findet RL in der Industrie bisher nur geringe Anwendung.

Immer kürzer werdende Produktzyklen und steigende Produktvielfalt stellen für die heutigen Produktionsprozesse eine große Herausforderung dar. Zukünftige Produktionen müssen immer anpassungsfähiger werden. Zeitgleich soll der Personalaufwand aufgrund des anhaltenden Fachkräftemangels möglichst gering ausfallen. Maschinelles Lernen, insbesondere das Reinforcement Learning, kann bei der Bewältigung dieser Herausforderungen eine relevante Rolle übernehmen.

Auch bei der Bekämpfung des Klimawandels kann Reinforcement Learning unterstützen. Um unsere Klimaziele zu erreichen, ohne unseren Lebensstandard signifikant zu senken, ist eine Optimierung des Ressourcenbedarfs nötig. Mit ausreichender Trainingszeit sind RL-Algorithmen sehr gut in der Optimierung von Prozessen und somit auch in dessen Ressourcenverbrauches. Google, als einer der Vorreiter im Gebiet des maschinellen Lerners, konnte durch ML-Algorithmen den Energieverbrauch der Kühlung ihrer Rechenzentren um bis zu 40 Prozent reduzieren.

Mittlerweile existiert eine Vielzahl an unterschiedlichen Reinforcement Learning Algorithmen. Während die mathematischen und strukturellen Unterschiede meist gut dokumentiert und einsehbar sind, ist ein direkter Vergleich der Leistungsfähigkeit der Algorithmen in verschiedenen Umgebungen nur schwer zu finden. Aus diesem Grund beschäftigt sich diese Ausarbeitung mit dem Vergleich von beliebten RL-Algorithmen anhand von Umgebungen mit geringer Komplexität.

Während die Zeit, welche ein RL-Algorithmus zum Lernen benötigt, bei der Anwendung auf Brett- und Computerspielen eher eine untergeordnete Rolle spielt, ist sie in der Anwendung in industriellen Umgebungen deutlich relevanter. Zum einen verlangsamen hohe Trainingszeiten den Entwicklungsprozess deutlich, was wiederum zu höheren Lohn- und Entwicklungskosten führt. Zum anderen ist es in vielen Anwendungsfällen nötigt, dass die Umgebung während des Trainingsprozesses dem Algorithmus zur Verfügung steht. Im Fall von Produktionsanlagen ist Trainingszeit somit sehr kostspielig. Aus diesem Grund wird neben der Leistungsfähigkeit auch die Lerngeschwindigkeit der Algorithmen im Folgenden untersucht.

2 Grundlagen

2.1 Definitionen

Um Reinforcement Learning im Folgenden besser beschreiben zu können, ist zunächst die Klärung einiger Grundbegriffe nötig. Diese sind aus der englischen Sprache entstanden, auf eine Übersetzung dieser Begriffe in das Deutsche wurde verzichtet, um eine Vergleichbarkeit zu anderen Werken in diesem Themenbereich zu gewährleisten.

1. Agent
   Der Agent ist die Instanz, welche Aktionen in einem Szenario/Umfeld ausführt und dafür eine Belohnung bekommt. Environment

2. Environment
   Das Environment ist, wie die deutsche Übersetzung schon vermutet lässt, die Umgebung in dem sich der Agent befindet. Das Environment legt dabei die grundlegenden Regeln fest und definiert, welche Aktionen möglich sind. Das Environment trägt somit ausschlaggebend zur Komplexität der zu lösenden Aufgabe bei. In vielen Fällen, so auch in den in dieser Ausarbeitung folgenden Versuchen, ist das Environment eine Simulation. Dies ermöglicht einen deutlich schnelleren Lernprozess, da jegliche Interaktion ohne nennenswerte Verzögerung ausgeführt werden kann. Bei komplexen Aufgabestellungen ist es so zudem möglich, mehrere Agenten parallel zu trainieren.

3. Action
   Als Action wird eine Interaktion des Agent mit dem Environment beschrieben. Die Lösung eines Problems kann somit als Abfolge bestimmter Actions angesehen werden. Welche Actions der Agent ausführen kann, hängt dabei von den Grundregeln des Environments ab.

4. State
   Der State ist der eindeutige und vollständige Beschreibung des Zustands, in welchem sich das Environment befindet. Aus technische Sicht ist der State meist ein Vektor, eine Matrix und ein Tensor, welcher alle relevanten Information des aktuellen Zustands enthält.

5. Reward
   Der Reward ist die unmittelbare Belohnung, welche der Agent als Feedback zu einer Action erhält. In der Praxis ist dies ein numerischer Wert, welche entweder erhöht oder reduziert werden kann. Der Agent kann so für eine Action belohnt oder bestraft werden, dabei versucht er sein Handeln so auszurichten, dass er die größte mögliche Belohnung erreicht. Die Art und Weise, wie der Reward vergeben wird, bestimmt somit das Verhalten des Agents.

6. Episode Als Episode wird ein vollständiger Durchlauf während des Trainings bezeichnet. Jede Episode startet mit dem Anfangszustand des Environments und kann auf mehreren Wegen enden. Im besten Fall wird die Episode beendet, weil die gestellte Aufgabe vom Agent gelöst worden ist. In vielen Fällen wird eine Episode jedoch abgebrochen, weil die maximale Anzahl an Actions überschritten wurde. Eine solche Grenze wird implementiert, um sicherzustellen, dass das Training effektiv und effizient abläuft. Wenn keine Obergrenze festgelegt wird, kann der Agent endlos versuchen, das Ziel zu erreichen, ohne jemals erfolgreich zu sein. Zudem kann so verhindert werden, dass der Agent in einer Schleife von kleinen Belohnungen feststeckt. Die letzte Möglichkeit, wie eine Episode enden kann, ist von Environment definiert, in vielen Fällen kann der Agent durch bestimmte Fehlentscheidungen die Episode beendet.

7. Bellmann Equation?
8. Markov Decision Process
9. Exploration vs Exploitation
10. Temporal difference vs Monte Carlo
11. Source

2.2 Algorithmen

2.2.1 Q-Learning

2.2.2 SARSA

3 Implementierung

3.1 Probleme

1. Taxi
   Für das Taxiproblem wurde ein fünf mal fünf großen Feld implementiert, der Agent (Taxifahrer) hat dabei die Möglichkeit, sich in alle vier Himmelsrichtungen zu bewegen. Die Aufgabe besteht darin, den Passagier, welcher sich in einer zufälligen Ecke des Feldes befindet, abzuholen und ihn dann zu seinem gewünschten Ziel zu bringen. Das gewünschte Ziel des Passagiers ist immer einer der verbleibenden Ecken des Spielfeldes. Um diese Aufgabe noch etwas zu erschweren, wurden zusätzlich noch Wände in das Feld mit integriert. Diese befinden sich immer an den gleichen Positionen und verhindern Bewegungen in bestimmte Richtungen.

   Für das Bewältigen dieser Aufgabe kann der Agent zu jedem Zeitpunkt zwischen sechs Actions entscheiden; Bewegung in jeweils einer der vier Himmelsrichtungen, aufnehmen des Passagiers und das Absetzen des Passagiers. Natürlich sind nicht alle Actions zu jedem Zeitpunkt sinnvoll.

   Der Zustand des Feldes kann durch 500 diskrete States beschrieben werden. Diese Anzahl ergibt sich aus der Multiplikation der 25 möglichen Position des Taxis, der fünf möglichen Positionen des Passagiers (beinhaltet den Fall, dass sich der Passagier im Taxi befindet), mit den vier möglichen Zielorten.

   Da das Abliefern des Passagiers an dem richtigen Strandort das Ziel der Aufgabe ist, bekommt der Agent dafür auch die höchste Belohnung. Damit die Erfüllung der Aufgabe so effizient wie möglich durchgeführt wird, bekommt der Agent für jede Action, die er durchführt und die nicht zu einer Belohnung führt, eine kleine Strafe. Zuletzt wird der Agent stark bestraft, wenn er den Passagier an dem falschen Ort absetzt oder versucht, einen Passagier an einer falschen Position aufzunehmen.

2. Cliff
   Wie für das Taxiproblem wurde auch für das Cliff Walking Problem ein Feld implementiert, dieses Mal allerdings in der Größe von vier mal zwölf. Die Aufgabe des Agenten besteht darin, von einer Seite des Feldes zur anderen zu gelangen, ohne dabei in die auf dem Feld befindliche Kippe zu fallen. Als Klippe wurden alle Felder definiert, welche sich auf dem direkten Weg zwischen Agent und Ziel befinden. Der Agent hat somit die Aufgabe, um diese Klippe herum zu navigieren und so das Ziel zu erreichen. Um die Herangehensweise der verschiedenen Algorithmen besser vergleichen zu können, befinden sich bei diesem Experiment alle Objekte (Agent, Klippen, Zielpunkt) zu Beginn jeder Episode an derselben Position.

Damit sich der Agent auf dem Feld bewegen kann, stehen ihm vier verschiedene Actions zur Verfügung, welche den Agent jeweils um ein Feld in einer der Himmelsrichtungen verschiebt.

Um den Zustand des Environments zu beschreiben, ist die aktuelle Position des Agent ausreichend. Insgesamt gibt es 48 (4×12) verschiedene Positionen auf dem Feld. Da das Betreten des als Kippe definierten Bereiches jedoch zum Ende der Episode führt, sind diese Positionen kein gültiger State. Gleiches gilt auch für die Zielposition. Bei zehn Klippen und einem Ziel ergeben sich so 37 States.

Neben des Erreichen des Ziels ist es besonders wichtig, dass der Agent nicht in die Klippe fällt, daher ist dies mit einer hohen Bestrafung für den Agent versehen. Zudem soll das Ziel so schnell wie möglich erreicht werden, daher ist, wie auch bei Taxi Problem, jede Action mit einer kleinen Strafe belegt. Eine explizite Belohnung des Agent ist für diesen Anwendungsfall nicht nötig, da das Ziel zur Beendung der Episode führt und das beste Ergebnis somit das ist, welches zur geringsten Bestrafung führt.

3. Frozen Lake
   Auch das Frozen Lake Problem ist auf einem Gitternetz aus Feldern implementiert. Einige Felder sind Eisplatten, die der Agent sicher betreten kann, während andere Felder Löcher darstellen, in die der Agent fallen kann und damit das Spiel verliert. Das Ziel des Agenten besteht darin, sicher zum Ziel zu gelangen, welches sich auf der anderen Seite des Sees (Gitternetz) befindet. Die besondere Herausforderung bei diesem Problem sind die Eisplatten, bewegt sich der Agent auf einer dieser Platten in eine bestimmte Richtung besteht eine Change, dass der ausrutscht und sich in eine andere Richtung bewegt.

   Wie auch beim Cliff Problem kann der Agent nur durch Bewegung mit dem Environment interagieren und hat somit vier Actions zur Verfügung.

   Eine weitere Herausforderung besteht darin, dass die Positionen der Löcher nicht fest sind, sondern bei Beginn jeder Episode zufällig festgelegt werden. Dies muss beim Abbilden des Zustandes des Environments als States berücksichtigt werden. Ein vier mal vier großes Environment mit drei Löchern hat somit 4368 mögliche States. Dies setzt sich zusammen aus den 12 möglichen Positionen des Agenten (16 Felder minus die Löcher und des Ziels) multipliziert mit den 364 möglichen Kombinationen, die 3 Löcher auf 14 freie Flächen zu verteilen.

   Die Struktur des Rewards ist für dieses Problem recht simpel, der Agent wird belohnt, wenn er das Ziel erreicht und bestraft, wenn er in ein Loch fällt.