REINFORCEMENT LEARNING

Katharina Strecker - ks188@hdm-stuttgart.de

Aktuelle Themen: Künstliche Intelligenz - WS 18/19

Die Entwicklungen und Erfolge im Bereich künstliche Intelligenz (KI) sind in den vergangenen Jahren exponentiell gestiegen und auch aus den Medien ist KI nicht mehr wegzudenken. Eine Technologie scheint in diesem Themenkomplex jedoch besonders herauszustechen und sorgt nicht nur für spektakuläre Schlagzeilen, sondern stellt auch einer der zukunftsweisenden Bereiche des maschinellen Lernens da: Reinforcement Learning.

Was Reinforcement Learning von anderen Lernmethoden unterscheidet, wie es bereits angewandt wird und welche Chancen und Herausforderungen in der Zukunft zu bewältigen sind, soll in diesem Artikel erläutert werden.

1 EINFÜHRUNG

Ein Großteil der Anwendungen im Bereich der KI nutzen Algorithmen aus dem Bereich des maschinellen Lernens bzw. Machine Learning (ML). Das allgemeine Prinzip von ML ist es, aus Erfahrungen (z.B. Daten) zu lernen und somit das Verhalten des Algorithmus zu verbessern [1]. Dieses Lernen findet oftmals in einem Trainingsprozess statt, durch das ein trainiertes Modell entsteht, welches dann wiederum auf neue, ungesehene Daten angewendet werden kann. Egal ob Gesichtserkennung, Windvorhersage oder intelligente Suchvorschläge, dem Erfolg dieser Anwendungen liegt häufig ein solcher Lernprozess zugrunde.

Wie der Algorithmus aus den Daten lernt, kann sehr unterschiedlich sein und ist auch stark von der zu lernenden Funktion abhängig. Allgemein kann im ML die Art eines Lernalgorithmus in drei Kategorien eingeteilt werden: Supervised Learning, Unsupervised Learning und Reinforcement Learning.[1]

Um die Unterscheidung von Reinforcement Learning zu Supervised und Unsupervised Learning zu verstehen, sollen diese beiden hier nochmal knapp zusammengefasst werden.

SUPERVISED LEARNING

Supervised Learning, also überwachtes Lernen, wird auch als Lernen mit Lehrer bezeichnet. Gelernt wird eine Funktion anhand Eingabedaten, die jeweils einer bestimmten Ausgabe (einem Funktionswert) zugeordnet sind. Ein typisches Beispiel wäre eine Klassifikation von Bildern. Beim Training eines neuronalen Netzes würde dieses als Eingabe

[1] Die Einteilung der Lernalgorithmen ist in der Literatur nicht einheitlich. In manchen Texten wird lediglich zwischen Unsupervised und Supervised Learning unterschieden.

ein Bild erhalten und als Ausgabe soll gegeben werden ob auf dem Bild ein Hund oder eine Katze zu sehen ist. Der Algorithmus würde sofort eine Rückmeldung erhalten, ob die berechnete Ausgabe falsch oder richtig ist, da jedes Bild mit dem richtigen Ausgabewert versehen ist (gelabelte Daten). Supervised Learning nennt man deshalb Lernen mit Lehrer, da es im Lernprozess stets eine Rückmeldung gibt und daraus gelernt werden kann.

UNSUPERVISED LEARNING

Beim Unsupervised Learning wird unüberwacht also ohne Lehrer gelernt. Konkret bedeutet das, dass dem Algorithmus nur Eingabedaten gegeben werden aber ohne die zugehörigen Ausgabewerte. Da der Algorithmus keine Rückmeldung über die Korrektheit der Ergebnisse bekommt, muss hier selbständig eine Struktur / ein Muster in den Daten gefunden werden. Ein typisches Beispiel ist das Clustering-Verfahren, bei dem der Algorithmus lernt ähnliche Daten zu erkennen und in Gruppen (Cluster) zusammenzufassen. Mit einem gut gelerntem Clustering wäre ebenfalls eine Unterscheidung zwischen Hunden und Katzen auf Bildern möglich. Häufig kommt Unsupervised Learning aber dann zum Einsatz, wenn das “labeln” (eindeutiges Zuordnen der Daten zu einem Ausgabewert) zu aufwendig oder nicht möglich ist. Zum Beispiel beim Einteilen von Kunden in Gruppen mit ähnlichen Vorlieben anhand Kundendaten. [2]

GRENZEN DES DATEN-BASIERTEN LERNENS

Supervised und Unsupervised Learning haben die Gemeinsamkeit, dass anhand (vieler) Daten gelernt wird. Da diese Daten sehr unterschiedlich sein können, ist es auch möglich beide Ansätze in sehr unterschiedlichen Anwendungsbereichen und für Problemlösungen verschiedenster Art einzusetzen. Nichtsdestotrotz gibt es Problemstellungen, bei denen das ausschließliche Lernen anhand Daten keine guten Ergebnisse erzielen kann.

Ein Beispiel:
Man möchte eine KI-Anwendung erstellen die in der Lage ist den Rubik’s Cube (Zauberwürfel) von beliebigen Startpositionen mit so wenigen Schritten wie nötig zu lösen.

Ein Ansatz dieses Problem mit Daten-basiertem Training zu lösen, könnte wie folgt aussehen: Die Input Daten geben einen Zustand bzw. die Verdrehung des Würfels an und die zugeordnete Ausgabe dieser Daten den Lösungsweg eines geübten Spielers. Bei einem solchen Ansatz gibt es allerdings zwei Probleme: Die erste Problematik ist, dass es über 43 Trillionen mögliche Ausgangszustände für den Würfel gibt. So ist es aus Zeit und Aufwandsgründen fast nicht möglich auch nur einen Bruchteil davon in den Trainingsdaten abzudecken. Das zweite und viel bedeutendere Problem ist, dass die zugeordnete Ausgabe nicht zwangsläufig den idealen und schnellsten Lösungsweg beschreibt. Durch Supervised Learning könnte das Netz nur Lösungswege berechnen, die maximal so gut sind wie die eines Menschen.

Mit Reinforcement Learning kann ein Problem wie das Lösen des Rubik’s Cube nicht nur überdurchschnittlich gut gemeistert werden, sondern dies auch ohne Verwendung von Trainingsdaten oder Strategie-Vorkenntnissen. [3]

2 KONZEPT REINFORCEMENT LEARNING

Reinforcement Learning beschreibt eine Lernmethode mit dem Ziel durch Bestärkung eine (optimale) Strategie für ein Problem zu finden [1]. Diese Bestärkung kann in Form von Belohnung oder negativer Belohnung (Bestrafung) stattfinden, die je nach getroffenen Aktionen vergeben werden können. Reinforcement Learning wird auch als Lernen mit Kritiker bezeichnet, da hier nicht wie beim Lernen mit Lehrer eine Rückmeldung gibt, ob die getroffene Entscheidung richtig oder falsch war, sondern nur eine Bewertung bzw. Kritik der vorherigen Aktionen vergeben wird. Meistens kann eine solche Kritik nur nach einer langen Sequenz von Aktionen geliefert werden.

Um Reinforcement Learning auf ein konkretes Problem anwenden zu können, wird dieses oft als sogenanntes Markov Decision Process formuliert. Dafür sollte das Problem in folgende Komponenten aufgeteilt werden:

Agent, Environment, State, Action und Reward.

Komponente	*Beschreibung*	*(Mögliche) Umsetzung im Beispiel RL Schachcomputer*
Agent	Der Entscheidungsträger innerhalb der Umgebung. Durch ihn wird entschieden welche Aktionen getroffen werden. Sein Ziel ist es dabei die maximale Belohnung zu erhalten.	Der (virtuelle) Spieler. Dieser entscheidet welcher Schachzug ausgeführt wird, mit dem Ziel das Spiel zu gewinnen.
Environment	Die Umgebung, in welcher der Lernprozess stattfindet. Diese definiert Zustände, mögliche Aktionen und wann Belohnungen vergeben werden.	Das Schachbrett und die Spielregeln. Diese geben z.B. vor, welche Züge welche Figur machen darf, wann das Spiel zu Ende ist und welche Belohnung dabei vergeben wird.
State	Der Zustand des Environments zu einem diskreten Zeitpunkt t.	Der Zustand des Schachbretts, der geschlagenen Figuren und welche Spieler am Zug ist.
Action	Das Resultat einer Entscheidung des Agents.	Ein konkreter Zug. Die Anzahl der möglichen Actions können, je nach State (also Stellungen der Figuren), in diesem Beispiel sehr unterschiedlich sein.
Reward	Das Feedback, das der Agent nach einer ausgeführten Aktion erhält. Dieses findet meistens nur selten statt und kann positiv (Belohnung) oder negativ (Bestrafung) sein. Der Reward kann auch unterschiedlich stark gewichtet sein, sodass kleine Erfolge einen	Könnte so definieren sein, dass ein kleiner positiver Reward vergeben wird, wenn der Agent eine Spielfigur des Gegners schlägt und ein kleiner negativer, wenn er eine Figur verliert. Gewinnt der Agent das komplette Spiel, könnte er dann einen

	kleineren Reward ausschütten als größere Erfolge. Den Reward für ein Problem zu definieren ist eines der schwierigeren Aufgaben bei RL und trägt viel zum Lernerfolg bei.	großen positiven Reward erhalten und einen großen negativen Reward für eine Niederlage.

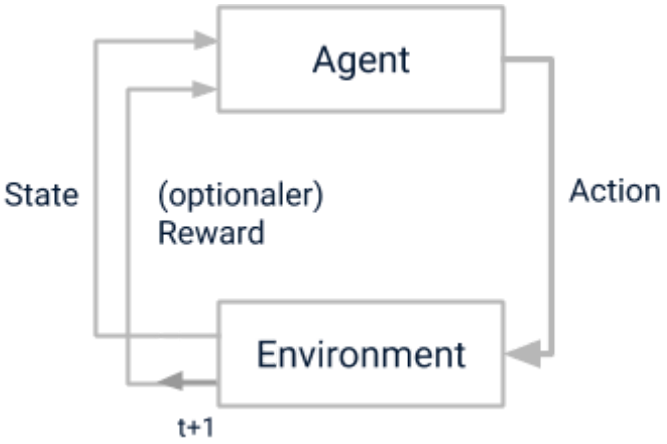

Abb.1: *Zusammenhang zwischen den Komponenten: Je nach ausgeführter Action, bestimmt das Environment im nächsten Zeitschritt (t +1) welche Reward vergeben wird und in welchem State der Agent nun eine erneute Action treffen muss.*

MARKOV DECISION PROCESS

Sind diese Komponenten definiert, kann das Problem als Markov Decision Process dargestellt werden. Der Markov Decision Process ist ein stochastischer Prozess, der nach dem russischen Mathematiker Andrei Andrejewitsch Markow benannt ist. Ziel bei der Anwendung von Markov-Prozessen ist es, die Wahrscheinlichkeiten für das Eintreten zukünftiger States und Rewards berechnen zu können. Dies geschieht unter der Annahme, dass diese Wahrscheinlichkeiten allein aus dem aktuellen State und der getroffenen Action berechnet werden können. [1]

In dem Beispiel des Schachcomputers sind die States die auf eine Action folgen zu jedem Zeitpunkt bekannt, da es sich um eine deterministisch und vollständig beobachtbare Umgebung handelt. Dieses Wissen ist aber nicht ausreichend für den Agenten damit dieser sinnvolle und zielführende Actions durchführen kann. Benötigt werden Belohnungswahrscheinlichkeiten, die durch State und Action wie folgt definiert werden:

(State s + Action a) = wahrscheinlicher Reward r

Der Agent soll nun eine Strategie lernen, welche die aufsummierte (kumulierte), maximale Belohnung ergibt. Dies wird häufig durch eine Belohnungsfunktion (Value Function) umgesetzt.

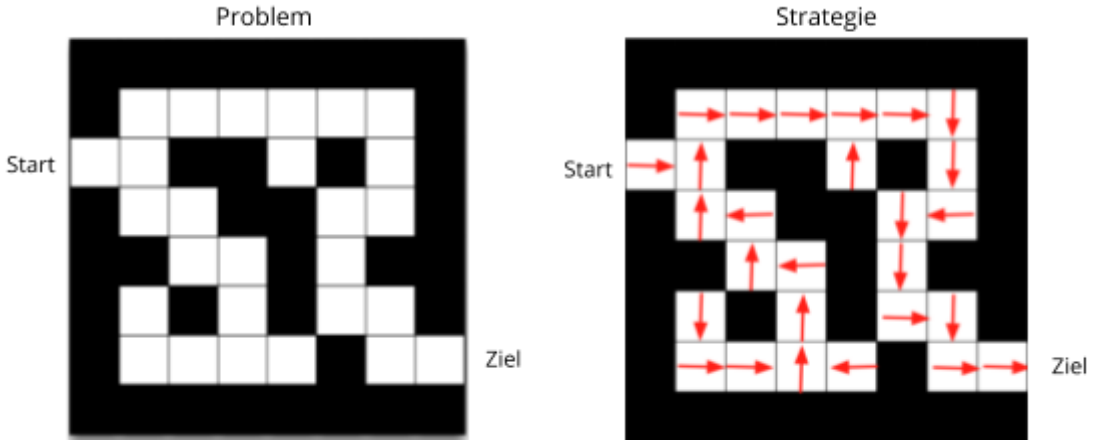

Abb.2: *Beispiel einer gelernten Strategie für die Lösung eines einfachen Labyrinths* [14]

Wie beim Supervised und Unsupervised Learning wird also wieder eine Funktion gelernt. Im Falle von Reinforcement Learning geschieht dieses Lernen dann meistens durch das "durchspielen" von vielen verschiedenen Situationen, woraufhin die Belohnungsfunktion immer wieder aktualisiert wird.

Am Anfang des Lernprozesses würde unser RL- Schachcomputer nur zufällige Aktionen treffen und sehr schnell Figuren verlieren und eine Niederlage erzielen. Man erhofft sich aber, dass nach mehreren Iterationen die Belohnungsfunktion so aktualisiert wird, dass bestimmte Actions in bestimmten States vermieden werden (z.B. König bewegen im ersten Teil des Spiels) und erste Erfolge stattfinden. Je länger das Training, desto genauer wird die Belohnungsfunktion und desto besser die Spielleistung, wobei auch hier, wie bei den anderen Lernmethoden, ab einem bestimmten Moment keine Verbesserung mehr stattfindet.

Abb.3: *Bei der Online Schachplattform Lichess[2] werden im Analyse-Spielmodus Bewertungen zu jedem Zug gegeben (Links im Bild). Ähnliche Wertungen soll auch der Agent zu jedem State-Action Paar lernen.*

[2] Mehr Informationen: https://lichess.org/analysis

Q-LEARNING

Wie schon erwähnt, stellt das Schach-Beispiel eine Besonderheit dar, da hier die Umgebung vollständig beobachtbar und folgende States und Rewards bekannt sind bzw. erlernt werden können. Die Art mit einer Value Function hier eine optimale Strategie zu finden nennt man auch modellbasierter Ansatz. [1]

In vielen Problemstellungen können Environment, States und Rewards aber gar nicht oder nur teilweise bestimmt werden. In einem Pokerspiel sind zum Beispiel die Spielkarten der Gegner unbekannt und welche Karten gezogen und gelegt werden ist nicht-deterministisch. Ähnlich ist es auch bei Computerspielen in denen oft Zufall eine Rolle spielt oder das Environment bzw. die States erst vom Agent erkundet werden müssen. In diesem Fall kann eine optimale Strategie ausschließlich auf Wahrscheinlichkeiten basieren, dass eine Action a auf einen State s folgt. Diese Wahrscheinlichkeiten wird auch als Bewertung oder Quality Q(s, a) für einen bestimmten Zustand genannt. Mit dem sogenannten Q-Learning Ansatz kann diese Bewertung dann gelernt werden.

DEEP REINFORCEMENT LEARNING

Da sich Reinforcement Learning häufig mit sehr komplexen Problemstellungen befasst, werden zum Berechnen der idealen Funktion oftmals tiefe neuronale Netze verwendet. Im Beispiel Q-Learning mit einem (Deep) Q-Network oder bei Suchbaum-basierten Funktionen mit Policy- und Value-Networks. [4] [5]

HERAUSFORDERUNGEN

In allen Lernmethoden von Machine Learning wird aus Erfahrungen gelernt. Was im Falle von Supervised und Unsupervised Learning die Daten sind, sind beim Reinforcement Learning die Rewards, welche aus unterschiedlichen State-Action Kombinationen in Erfahrung gebracht werden können. Wann und in welcher Höhe ein Reward vergeben werden muss ist oftmals aber sehr schwierig zu definieren. Besonders dann, wenn es ein primäres Ziel gibt, das nur schwer in Unterziele aufgeteilt werden kann. Möchte man beispielsweise einem Roboterarm mit Reinforcement Learning beibringen einen Gegenstand erfolgreich zu greifen, ist ein Unterziel schwer definierbar. Da die Auswahl der möglichen Actions (z.B. mehrere Bewegungsmöglichkeiten pro Gelenk) sehr groß ist und zu Beginn des Trainings oft willkürlich Actions durchgeführt werden, kann es sein, dass in berechenbarer Zeit keine zielführende Bewegung durchgeführt werden würde. Durch die fehlenden Rewards wird diese Problematik noch verstärkt. [6]

Eine weitere Schwierigkeit, die hier ebenfalls angesprochen wurde, sind lange Trainingszeiten. Der State eines Schachbretts lässt sich zum Beispiel mit wenigen Zahlen darstellen, was das Training von verschiedenen Spielzügen sehr einfach gestaltet. Viele Probleme benötigen aber einen visuellen Input, wie es zum Beispiel bei den Atari 2600 Spielen der Fall ist. Umso größer diese Daten und die Berechnungsdauer eines States, desto immens länger der Trainingsprozess. Bei sehr komplexen Anwendungen können so, selbst mit der besten Hardware, mehrere Monate an Training anfallen. [7]

3 ERFOLGE UND ANWENDUNGEN

Besonders in den vergangenen fünf Jahren wurden zahlreiche Anwendungen basierend auf Reinforcement Learning entwickelt und mit Forschungsprojekten neue Durchbrüche erzielt. Ein Überblick über diese Erfolge soll hier jeweils in den Kategorien Gaming, Robotics und 3D gegeben werden.

GAMING

Die größten und spektakulärsten Erfolge hat Reinforcement Learning vor allem in der Gaming-Branche erzielt. Grund dafür ist sicherlich, dass hier das Finden einer geeigneten Strategie eine zentrale Rolle spielt, was gleichzeitig die oftmals stark regelbasierte Umgebung erleichtert. Besonders wichtige Rollen spielten bei den Erfolgen das Unternehmen DeepMind und die Non-Profit-Organisation OpenAI.

Q-Networks und Atari 2600

Den Auftakt in eine neue Ära im Reinforcement Learning brachte DeepMind mit ihrer Veröffentlichung *"Human-level control through deep reinforcement learning"* im Frühjahr 2015. Hier stellte DeepMind Q-Learning durch tiefe Neuronalen Netzen (Q-Networks) vor, mit denen es ihnen gelang eine Vielzahl an Atari 2600 Spielen mit nur einem Netzwerk auf überdurchschnittlich hohem Niveau zu spielen. Die Besonderheit dabei war nicht nur, dass das Netzwerk lediglich auf Basis von Pixel und Score-Daten die Spiel-Strategien lernt, sondern dass hier eine Art Transferlernen von einem Spiel auf das andere stattfand. [8]

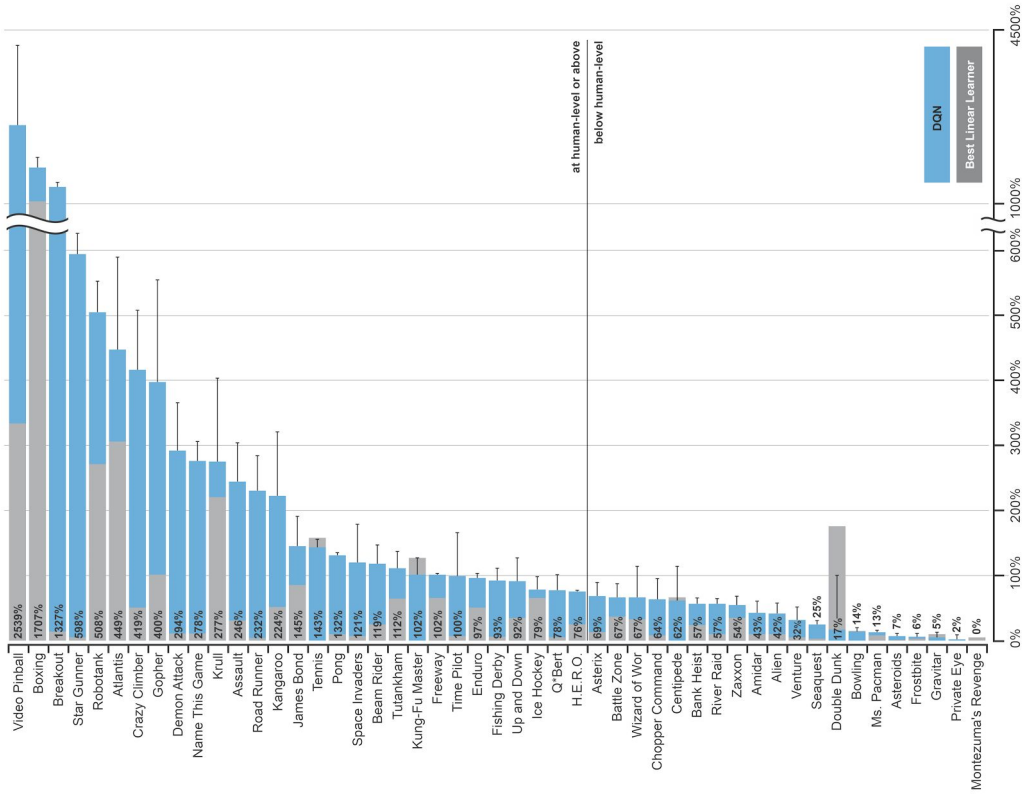

Abb.4: *Überblick der Leistung des gelernten Q-Networks bei unterschiedlichen Atari 2600 Spielen*

Alpha Go, Alpha Go Zero & Alpha Zero

Auch in der nicht-digitalen Spielewelt erzielte DeepMind neue Weltrekorde. 2014 schaffte es ihr Programm AlphaGo als erstes Computerprogramm einen professionellen Go-Spieler zu besiegen. Go unterscheidet sich mit ungefähr 200 durchschnittlichen Zugmöglichkeiten deutlich von anderen strategischen Brettspielen. Das Berechnen eines geeigneter Zugs mit Hilfe eines Suchbaums, wie es oftmals bei Schachprogrammen der Fall ist, ist für die Vielzahl der Zugmöglichkeiten bei Go nicht mehr anwendbar. DeepMind erschaffte hier mit ihrem neuen Ansatz von Monte Carlo Tree Search in Kombination mit zwei neuronalen Netzen (Policy/ Value) eine intelligente, Reinforcement Learning basierte Lösung. [9]

Zwei Jahre später im Oktober 2017, erfolgte ein weiterer Rekord im Bereich Go-Software. Mit **AlphaGo Zero** besiegte DeepMind nicht nur den eigenen Vorgänger AlphaGo, sondern das mit einer Trainingszeit von nur drei Tagen.

Zusätzlicher Unterschied zwischen diesen beiden Programmen war, dass AlphaGo Zero ausschließlich durch Spiele gegen sich selbst trainierte, während AlphaGo noch Profi-Spielzüge zum Lernen erhalten hatte. [10]

Zwei Monate später veröffentlichte DeepMind eine noch bessere Go-Software: **AlphaZero**. Diese besiegte nicht nur ihren Vorgänger nach ebenfalls drei Tagen Trainingszeit, sondern wurde zusätzlich noch für weitere Spiele generalisiert. So ist sie in der Lage, sowohl die derzeit beste Schachsoftware (Stockfish) als auch Shogysoftware (Elmo) zu besiegen. Mit dieser Generalisierung zeigte DeepMind erneut das Potential von Reinforcement Learning. [11]

Dota 2 Multiplayer mit OpenAI Five

Dass Reinforcement Learning auch die Komplexität eines Multiplayer Echtzeit-Strategiespiels bewältigen kann, bewies OpenAI mit ihrer Software OpenAI Five im Sommer 2018.

In 5v5 Partien gegen sowohl Amateur- als auch semi-professionelle Spieler, konnte die Software trotz der Schnelligkeit und Unberechenbarkeit der Gegner mehrere Spiele gewinnen. Auch wenn der Erfolg gegen ein professionelles Team ausblieb, zeigte OpenAI Five dennoch, dass die Forschung für Reinforcement Learning in Multiplayer-Umgebungen ebenfalls großes Potential zeigt.

Das System wurde dabei mehrere Monate durch das Spielen gegen sich selbst trainiert. Laut OpenAI gleicht ein Tag Training einer tatsächlichen Spielzeit von 180 Jahren. [7]

Weitere Anwendungen und Erfolge

Auch zahlreiche andere Spiele wurden in den vergangenen Jahren mit Hilfe von Reinforcement Learning gemeistert. Darunter viele 2D Spiele wie Super Mario World[3], Flappy-Bird[4] oder Pac-Man[5]. Aber auch in anderen Multiplayer 3D Spielen wie Quake III

[3] Mehr Informationen: https://www.youtube.com/watch?v=qv6UVOQ0F44
[4] Mehr Informationen: https://github.com/yenchenlin/DeepLearningFlappyBird
[5] Mehr Informationen: https://github.com/tychovdo/PacmanDQN

Arena Capture the Flag[6] oder dem komplexen Echtzeit-Strategiespiel StarCraft II[7] konnten Erfolge gefeiert werden.

ROBOTICS

Künstlichen Intelligenz und Robotics sind schon seit vielen Jahren ein großes Thema in dem auch Reinforcement Learning zur Weiterentwicklung beiträgt. Da Roboter in der realen Welt agieren, die auch ungeplante und zufällige Situationen hervorrufen kann, ist hier ein selbstlernendes, Problemlösungs-orientiertes Verfahren besonders wichtig. [12]

Bewältigen alltäglicher Situationen durch Lernen von Geschicklichkeit

Was für uns Menschen alltäglich und einfach erscheint, kann für einen Roboter eine große Schwierigkeit darstellen, da viele Aufgaben besonderes Geschick benötigen.

Bei dem Öffnen einer Türklinke zum Beispiel, benötigt es sowohl das richtige Maß an Kraftaufwand als auch eine gut gesteuerte Bewegung der Hand. Wie dieses Problem für Roboter mit Reinforcement Learning zu lösen ist, zeigt DeepMind mit ihrer Veröffentlichung *"Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates"*[8]. In dem entsprechendem Video[9] zeigen sie, wie mehrere Roboterarme durch gleichzeitiges trainieren gegenseitig voneinander lernen können.

Dass das Wenden von Pfannkuchen auch für Roboter erst gelernt werden muss, zeigen Wissenschaftler vom Italian Institute of Technology in einem Video[10] zu ihrem Paper. Hier wird dem Roboterarm die richtige Durchführung zunächst von einem Menschen "demonstriert" und durch Reinforcement Learning ist es ihm dann möglich nach 50 Versuchen diese Bewegung zu beherrschen.

Umgang mit ungewohnten Situationen

Besonders wenn sich ein Roboter sich im Raum bewegen oder mit Menschen agieren muss, entstehen schnell unbekannte Situationen auf die gekonnt reagiert werden müssen. Ein gutes Beispiel aus der Kombination von beiden Unberechenbarkeiten zeigt das Massachusetts Institute of Technology mit Ihrem Video zu "*Socially Aware Motion Planning with Deep Reinforcement Learning*"[11]. Hier hat ein kleiner,fahrbarer Roboter die Aufgabe so durch ein Gebäude zum Ziel zu fahren, dass er eine Kollision mit Gegenständen oder Fußgängern vermeidet. Die Studie zeigt gut wie verschiedene Bereiche aus dem Feld KI, wie zum Beispiel Objekterkennung und Reinforcement Learning, zusammenspielen können.

[6] Mehr Informationen: https://deepmind.com/research/publications/capture-the-flag/
[7] Mehr Informationen: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/
[8] Mehr Informationen: https://deepmind.com/research/publications/deep-reinforcement-learning-robotic-manipulation/
[9] Mehr Informationen: https://www.youtube.com/watch?v=ZhsEKTo7V04
[10] Mehr Informationen: https://www.youtube.com/watch?v=W_gxLKSsSIE
[11] Mehr Informationen: https://www.youtube.com/watch?v=CK1szio7PyA

3D

Ein entscheidender Vorteil den die 3D-Anwendung von Reinforcement Learning gegenüber realem Echtzeit-Training hat, ist die deutliche Reduzierung von Trainingszeiten. Was in der Robotics in Echtzeit trainiert werden müsste und mehrere Stunden dauern würde, wird digital in wenigen Sekunden trainiert. Hinzu kommen natürlich die Kostenersparnisse, wenn teure Hardware und andere Materialien zum Einsatz kommen müssen.

OpenAI zeigt, wie Geschicklichkeit mit Reinforcement Learning in der virtuellen Simulation gelernt wird und daraufhin auf reale Roboter-Hände angewendet werden kann.[12]

Natürlich können hier gelernte Fähigkeiten aber auch für die 3D-Anwendung selbst verwendet werden, zum Beispiel in Spielen oder Filmen.

Bewegungen Lernen

Auch in der 3D Anwendung kann Reinforcement Learning bei der Erlernung neuer Bewegungsabläufe helfen. Mit dem Ziel so schnell wie möglich vorwärts zu kommen, zeigt DeepMind wie gut und gleichzeitig amüsant virtuelle Körper diese Aufgabe bewältigen können[13]. Auch selbstgelernter Zweikampf[14] und das Anziehen von Kleidung[15] löst DeepMind so mit Reinforcement Learning.

Das Unternehmen DeepMotion setzt ebenfalls auf RL, mit dem Ziel individuellere und situationsbasiertere Bewegungen im 3D-Bereich zu erzeugen. Ein Beispiel wäre die Erlernung von verbessertem Dribbling bei einem virtuellem Basketball-Spieler[16].

WEITERE ANWENDUNGSBEREICHE

Auch wenn die Gaming-, Robotics- und 3D-Branche den größten Teil der Reinforcement Learning Anwendungen ausmachen, gibt es auch weitere Bereiche, in denen die Lernmethode Verbesserungen verspricht.

Autonomes Fahren

Dass autonomes Fahren nicht eine riesen Menge an Technologien benötigt, wie bei den meisten Ansätzen, möchte das Unternehmen Wayve zeigen. Mit Reinforcement Learning und Computer Vision soll das Auto hier im Prinzip ganz von selbst das Fahren lernen.[17] Ein ähnliches Forschungsgebiet ermöglicht nun auch Amazon mit ihrem DeepRacer Projekt.[18]

Aktienhandel

Wer beim An- und Verkauf von Aktien Gewinne erzielen möchte, benötigt langfristig eine gute Strategie. Aus diesem Grund ist es naheliegend, dass Reinforcement Learning hier

[12] Mehr Informationen: https://openai.com/blog/learning-dexterity/
[13] Mehr Informationen: https://www.youtube.com/watch?v=hx_bgoTF7bs
[14] Mehr Informationen: https://sites.google.com/view/multi-agent-competition
[15] Mehr Informationen: https://www.youtube.com/watch?v=ixmE5nt2o88
[16] Mehr Informationen: https://blog.deepmotion.com/2018/08/07/deepdribble-simulating-basketball-with-ai
[17] Mehr Informationen: https://wayve.ai/about
[18] Mehr Informationen: https://aws.amazon.com/deepracer/

ebenfalls zu Einsatz kommt um eine solche Strategie zu erlernen. Start-Ups wie Necotic[19] machen sich daher die Lernmethode zu nutze.

4 ZUSAMMENFASSUNG UND AUSBLICK

Reinforcement Learning ist eine Lernmethode, die es in den vergangenen Jahren geschafft hat ungeahnte Lösungen für komplexe Probleme zu finden. Durch die Unabhängigkeit von großen Datensätzen und das Strategie-basierte Vorgehen, ergänzt es Supervised und Unsupervised Learning in vielen Anwendungsbereichen. Darüber hinaus ist es mit RL möglich ganz neue Lösungswege zu zeigen. Selbst bei Go, einem Spiel, das hunderte von Jahre Erfahrungen mit sich bringt, konnten Profispieler von DeepMinds Alpha Zero noch neue Strategien lernen. [13]

Ein weiteres Potential von Reinforcement Learning liegt in der Generalisierung des Lernens. Was DeepMind mit den Atari 2600 Spielen und Alpha Zero zeigt, ist, dass hier nicht nur ein konkretes Problem gelöst werden kann, sondern viele unterschiedlicher Art mit nur einem trainierten Model. Diese Erfolge machen die Idee von Artificial General Intelligence noch greifbarer.

Trotz der Herausforderungen im Design und Training von Reinforcement Learning Modellen, zeigen die Entwicklungen der letzten Jahre, dass die Technologie noch sehr großes Wachstumspotenzial hat. Besonders durch den geeigneten Einsatz von Reinforcement Learning in der Robotics-Branche müssen wir aber ein Auge auf den richtigen Verlauf dieser Entwicklungen behalten.

Sicher ist aber, dass weitere Erfolge von den Technologie-Vorreitern DeepMind und OpenAI nicht lange auf sich warten lassen werden.

[19] Mehr Informationen: https://neotic.ai/

QUELLEN

[1] J. Frochte, *Maschinelles Lernen: Grundlagen und Algorithmen in Python*. Carl Hanser Verlag GmbH Co KG, 2019.

[2] A. C. Müller and S. Guido, *Einführung in Machine Learning mit Python: Praxiswissen Data Science*. 2017.

[3] S. McAleer, F. Agostinelli, A. Shmakov, and P. Baldi, "Solving the Rubik's Cube Without Human Knowledge," 18.05.2018. [Online]. URL: https://arxiv.org/abs/1805.07470 [Letzter Zugriff: 15.03.2019].

[4] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[5] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[6] A. Gudimella *et al.*, "Deep Reinforcement Learning for Dexterous Manipulation with Concept Networks," 20.09.2017. [Online]. URL: https://arxiv.org/abs/1709.06977 [Letzter Zugriff: 15.03.2019].

[7] "OpenAI Five." [Online]. URL: https://blog.openai.com/openai-five/. [Letzter Zugriff: 15.03.2019].

[8] "Deep Reinforcement Learning," *DeepMind*. [Online]. URL: https://deepmind.com/blog/deep-reinforcement-learning/ [Letzter Zugriff: 15.03.2019]

[9] "AlphaGo: Mastering the ancient game of Go with Machine Learning" [Online]. URL: https://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html. [Letzter Zugriff: 15.03.2019].

[10] "AlphaGo Zero: Learning from scratch | DeepMind," *DeepMind*. [Online]. URL: https://deepmind.com/blog/alphago-zero-learning-scratch/. [Letzter Zugriff: 15.03.2019].

[11] D. Silver *et al.*, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," 05.12.2017. [Online]. URL: https://arxiv.org/abs/1712.01815 [Letzter Zugriff: 15.03.2019].

[12] P. Kormushev, S. Calinon, and D. Caldwell, "Reinforcement Learning in Robotics: Applications and Real-World Challenges," *Robotics*, vol. 2, no. 3. pp. 122–148, 2013.

[13] "AlphaZero: Shedding new light on the grand games of chess, shogi and Go." [Online]. URL: https://deepmind.com/blog/alphazero-shedding-new-light-grand-games-chess-shogi-and-go/. [Letzter Zugriff: 15.03.2019].

[14] "Advanced Topics 2015 Reinforcement Learning." [Online]. URL: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Letzter Zugriff: 15.03.2019].