DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Reinforcement Learning

Robotik II

für die Prüfung zum

Bachelor of Engineering

des Studienganges Informationstechnik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Jan Gerber

Abgabedatum 3. Mai 2017

Matrikelnummer	5757291
Kurs	TINF14B3
Gutachter der Dualen Hochschule	Prof. Dr. Marcus Strand

Erklärung

gemäß § 5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.
Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ort Datum

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Bei Reinforcement Learning handelt sich um eine Art von Machine Learning und ist damit auch ein Teilgebiet der Künstlichen Intelligenz. Mit Hilfe von Reinforcement Learning können Maschinen und Agents automatisch feststellen mit welchen Verhalten sie ihre Belohnungen maximieren können. Für die Feststellung der besten Strategie ist jegliche ein Feedback in Form einer Belohnung notwendig.

In der heutigen Zeit nimmt die Anzahl der Roboter in unsere Umwelt ständig zu. Angefangen bei einfachen Staubsaugrobotern bis zu allgemeinen Transport Robotern. Die meisten dieser Roboter sind von Menschen entwickelt und programmiert. Durch die feste Verdrahtung von Abläufen kann sich der Roboter nur schwer an neue Situationen und Umweltbedingungen anpassen. Hier kommt kommen nun Methoden von maschinellen Lernens ins Spiel. Mit Hilfe dieser Methoden können Roboter automatisch relevante Daten aus der Umwelt extrahieren und ihre Aufgabe meistern. Durch moderne Methoden wie Reinforcement Learning kann diese Aufgaben weiter automatisiert werden und die Lücke zur vollständig autonomen Robotern geschlossen werden. Diese könnten dann als allgemeine Hilfen im Haushalt, Versorgung älterer Menschen und allgemeine Aufgaben eingesetzt werden.

Geschichte Reinforcement Learning Reinforcement Learning entstand aus zwei Zweigen, welche sich getrennt entwickelt haben [SB98, S. 16f]. Der erste Zweig entstand aus der Psychologe von tierischem Lernen, der sich hauptsächlich mit dem Prinzip "Trial and Error"bei Tieren beschäftigt. Dieser Zweig beschäftigte sich zudem mit den Anfängen von künstlicher Intelligenz und führte Anfangs der 1980er zu der Entstehung von Reinforcement Learning. Der andere zweite Zweig entstand aus dem Problem einer optimalen Steuerung und ihrer Lösung mit Hilfe dynamischer Programmierung und Optimierungsfunktionen. Aus diesen beiden Zweigen entwickelte sich Ende der 1980er der moderne Ansatz des Reinforcement Learning.

Kapitel 2

Grundlagen

In diesem Kapitel wird auf die Grundlagen von Reinforcement Learning und auf das Markovsche Entscheidungsproblem eingegangen.

2.1 Einführung

Bei Reinforcement Learning geht um einen Agent, der die beste Methode / Strategie (Policy) versucht zu finden, um in der Umgebung (Environment) ein definiertes Ziel zu erreichen. Hierfür kann der Agent durch verschiede Aktionen von einem zum anderen Zustand wechseln bis er das gewünschte Ziel erreicht hat. Dabei soll der Agent versuchen durch selbständig herausgefundene Aktionsfolgen seine Belohnung zu maximieren. Bei der Belohnung kommt es nicht auf die momentane Belohnung an, sondern auch auf die späteren Belohnungen. Reinforcement Learning unterscheidet sich im Gegensatz zum überwachten Lernen dahingegen, dass beim überwachten Lernen für jede Situation eine Sollaktion

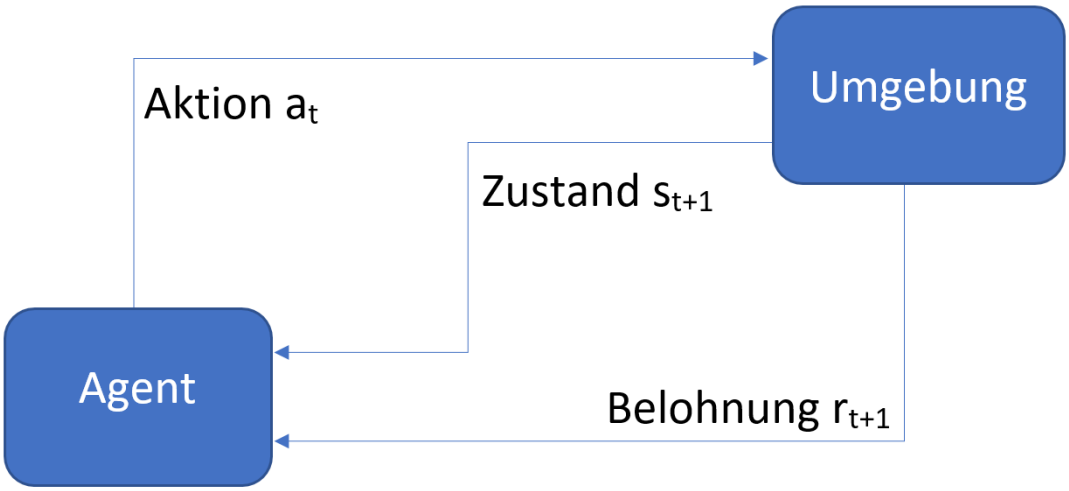

Abbildung 2.1: Reinforcement Learning Modell

gegeben ist. Beim Reinforcement Learning hat der Agent jedoch nur eine Belohnung als Rückmeldung gegeben.

Folgende Elemente sind außer dem Agenten und der Umgebung noch wichtig:

- **Policy** Eine Policy bestimmt das Verhalten des Agenten. Dabei können Policies zum Beispiel Lookup-Tabellen sein. Zusätzlich können Policies auch stochastische Abbildungen sein.
- **Reward Funktion** Mit Hilfe der Reward/Belohnungs-Funktion wird definiert, welches gute und schlechte Zustände sind. Dabei wird ein Zustand bzw. ein Aktion-Zustand Paar auf eine Zahl abgebildet. Reward-Funktionen können stochastisch sein.
- **Value Funktion** Die Value Funktion berechnet die erwartete zukünftige Belohnung, ausgehend vom derzeitigen Zustand. Da sich die Value-Funktion aus der Reward Funktion ableitet wird zunächst die aktuelle Belohnung benötigt.
- **Modell** Mit Hilfe eines Modells können Vorhersagen zum nächsten Zustand getroffen werden. Damit kann bereits im Vorhinein die erwartete Belohnung errechnet werden.

2.2 Markov Decision Problem

Der Ansatz von Reinforcemnt Learning geht davon aus das es sich bei RL um ein Markov Decision Problem handelt. Ein MEP ist ein 5 Tupel (S, A, T, R , γ):

- **S** Menge von Zuständen
- **A** Menge von Aktionen
- **T** Transitionswahrscheinlichkeit
- **R** erwartete Belohnung
- $\gamma \in [0, 1]$ Diskontierungsfaktor;
  - $\gamma \to 0$ nur sofortige Belohnung betrachten
  - $\gamma \to 1$ spätere Belohnung stärker berüksichtigen

Das Ziel des MDP ist es, eine Policy π zu finden, die den Gewinn/Belohnung maximiert. Dabei gilt die Markov Eigenschaft, dass der nächste Zustand s‘ und Belohnung nur vom vorherigen Zustand s und der durchgeführten Aktion a abhängen. Dabei spielen weder die vorherigen Zustände noch die vorherigen durchgeführten Aktionen eine Rolle.

2.3 Exploitation und Exploration

Exploitation beschreibt das Nutzen des aktuellen Wissens über die Umgebung, um eine möglichst gute Aktion zu wählen. Durch das Nutzen von nur bekannten Policies kann keine bessere und optimalere Policy gefunden werden. Exploration beschreibt das Wählen einer nicht optimalen / besten Aktion, damit herausgefunden werden kann ob die neue Aktion besser ist als die vorherig genutzte Aktion. Durch vermehrte Exploration kann sich die Performance hinsichtlich Zeit und Kosten, verschlechtert werden. Es besteht aber auch die Möglichkeit neuere bessere Aktionen zu finden. Da beide Methoden Vor- und Nachteile aufweisen, werden oft auch Kombinationen aus beiden Methoden genutzt.

2.4 Klassifikation von Lösungsverfahren

Die Lösungsverfahren können wie folgt Klassifiziert werden [Shi]:

- **Indirektes Lernen**: Durch das Schätzten eines expliziten Modells der Umgebung kann eine optimale Policy errechnet werden.
- **Direktes Lernen**: Die optimale Policy wird gelernt, ohne dass ein explizites Modell der Umgebung vorhanden ist. Diese Verfahren lassen sich zusätzlich in zwei unterschiedliche Kategorien unterteilen.
  - Policy-Search basierenden Methode
  - Value-Funktion basierende Methoden

Kapitel 3

Lösungsverfahren

Im folgenden Kapitel werden verschiedene Lösungsverfahren vorgestellt. Es wird dabei auf die in Kapitel 2.4 klassifizierten Lösungsverfahren mit dem Value-Funktion und Policy Search Ansatz eingegangen (entnommen aus [Hei+07]).

3.1 Value Function

Der Value Function Ansatz beschäftigt sich mit dem Finden einer optimalen Value-Function. Mit Hilfe dieser kann dann die optimale Policy gefunden werden. Beim Value-Function Ansatz handelt es sich um ein Optimalitätsproblem, es muss hierfür eine Lösung für die Formel 3.1gefunden werden.

$$V^*(s) = \max_{a \in A} \left(\sum_{s' \in S} T(s, a, s')(R(s') + \gamma V^*(s')) \right) \tag{3.1}$$

Dieses Formel entspricht dem Optimalitätsprinzip von Bellman. Eine optimale Policy hat die Eigenschaft, dass mit einer optimalen Policy immer die optimale Entscheidung getroffen wird egal was der initialer Zustand nach der ersten Entscheidung war.

In den folgenden Abschnitten werden zwei Ansätze vorgestellt.

Dynamic Programming Ein Ansatz ist die Lösung des Problems mit Hilfe von dynamic Programming. Bei diesem Ansatz handelt es sich um ein Modell basierter Ansatz. Eine typische Methode ist die Policy Iteration bei der zwischen zwei Phasen gewechselt wird. Zunächst wird die Value-Function für die jetztige Policy bestimmt. Nachdem die Value-Function bestimmt wurde wird für jeden Zustand die beste Aktion ausgewählt.

Temporal Difference Learning Beim Temporal Differnce Learning handelt es um ein Modell freier Ansatz. Dieser Ansatz passt seine Policy nicht erst an nachdem er die Belohnung erhält, sondern nach jeder durchgeführten Aktion, anhand der erwarteten

Belohnung. Hierfür wird mit Action-Value Funktion (siehe Formel 3.2) der Nutzen einer Aktion berechnet.

$$Q^{\pi}(s,a) = \sum_{s' \in S} T(s,a,s')(R(s') + \gamma V^{\pi}(s')) \tag{3.2}$$

Ein Beispiel für diese Methode ist der Q-Learning Algorithmus. Der Algorithmus beginnt damit, dass im aktuellen Zustand s ein Aktion a durchgeführt wird und wir eine Belohnung r erhalten. Aus dem Folgezustand s' wird die beste Aktion a' gewählt, anhand der besten Belohnung die mit Hilfe der Bewertungsfunktion bestimmt wurde. Die neue Funktion wird dann nach folgender Formel angepasst (siehe Formel 3.3). Wobei α der Lernrate entspricht, die mit der Zeit abnimmt. Es handelt sich dabei um ein off-Policy-Algorithmus.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{3.3}$$

3.2 Policy Search

Im Gegensatz zum Value-Function Ansatz wird beim Policy Search Ansatz direkt im Policy Raumnach der optimalsten Policy gesucht. Typischerweise sind die Policies in Klassen unterteilt. Diese sind dann wiederum anhand eines Parameter Vektor θ parametrisiert. Mit Hilfe des Vektors θ lässt sich die Suchkomplexität reduzieren, da bereits bekanntes Wissen eingebunden werden kann. Dieses Feature ist vor allem in der Robotik wichtig, da Wissen eines 'Experten' eingebunden werden kann. Policy-Search ist jedoch schwieriger, da die optimale Lösung meist nicht direkt mit Formeln gelöst werden kann. Die meisten Policy-Search Algorithmen optimieren deshalb bestehende Policies.

Kapitel 4

Herausforderung in der Robotik

Die Robotik bietet eine besondere Herausforderung an das Reinforcement Learning. Zunächst handelt es sich bei Robotern um kontinuierliche Systeme, sodass festgelegt werden muss in welchen zeitlichen Abständen der Roboter angesteuert wird. Bei einem Roboter handelt sich um ein komplexes physikalisches System, das eine lange Ausführungsdauer von Task besitzt, manuelle Eingriffe in das System benötigt, sowie Wartungen und Reparaturen hat. Kober J., Bagnell A. und Peters J. stellen im Artikel folgende Probleme in der Robotik im Bezug zu Reinforcement Learning vor[KBP13]:

4.1 Mehrdimensionalität

Der Begriff „Fluch der Dimensionalitäten“ wurde durch Bellman geprägt. Dieser beschäftigte sich mit der optimalen Kontrolle in diskreten, hochdimensionalen Räumen. Durch das Wachstum der Dimensionen nimmt die Anzahl der Daten und die benötigte Rechenleistung exponentiell zu.

Das Problem ist, dass Robotersysteme oft viele Dimensionen haben, da moderne Roboter eine hohe Anzahl an Degrees of Freedoms besitzen. Nehmen wir als Beispiel einen Roboterarm mit einer Degree-of-Freedom von sieben. Dabei würde eine Repräsentation eines Zustandes aus allen Winkeln und Geschwindigkeiten aller sieben Freiheitsgrade, sowie die kartesische Position und die Geschwindigkeit des Endeffektors bestehen. Das bedeutet, dass der Roboterarm aus $2 * (7 + 3) = 20$ Zuständen besteht. Wenn wir annehmen, dass jede Dimension in zehn Ebenen unterteilt ist, ergeben sich insgesamt 10^{20} unterschiedliche Zustände.

In Bereich von Reinforcement Learning wird deshalb in der Robotik mit Abstraktionen gearbeitet. Dies kann mit Hilfe von Diskretisierung von Objekten, Näherungsfunktionen und planen von Makro. Aufgaben erfolgen. Zusätzlich werden große Aufgaben in mehrere elementare Aufgaben zerkleinert.

4.2 Probleme der realen Umgebung

Da Roboter in einer realen Umgebung agieren und das Equipment meist sehr kostspielig, schwer zu warten und reparieren ist, muss auf die Hardware des Roboters, sowie die Umgebung des Roboters besonders achtgegeben werden. Dies bezieht sich vor allem auf die sichere Exploration von neuen Aktionen. Diese kann teilweise mit simulierten Umgebungen gemeistert werden.

Die Eigenschaften eines Roboters können sich je nach externen Faktoren stark unterscheiden. Faktoren wie Temperatur und Lichtverhältnisse können die Performance von Sensoren beeinflussen und haben damit direkten Einfluss auf das Ergebnis der Aktionen. Dies erschwert das Vergleichen verschiedener Algorithmen zusätzlich.

Viele durchgeführte Experimente mit Robotern benötigen zusätzlich menschliche Hilfe um wieder in den Startzustand zu gelangen. Zusätzlich besteht in dynamischen Umgebungen eine besondere Herausforderung, beispielsweise müssen bei fliegenden Robotern mehrere Sequenzen wiederholt werden bevor man sich im zu erlernenden Task wieder befindet. Bei dynamischen Task besteht nicht die Möglichkeit während eines Zustandswechsel die gesamte Umgebung zu pausieren um die neue Aktion zu berechnen und planen. Moderne effiziente Algorithmen müssen in der Lage sein innerhalb kürzester Zeit und wenig Versuchen das gewünschte Ergebnis zu erzielen, um Zeit, Arbeit und Kosten einzusparen.

In physischen Systemen haben wir zudem eine Verzögerung in der Sensorik und Aktorik. So repräsentieren die Sensordaten nicht die aktuellen Gegebenheiten des Roboters wieder. Motoren und andere Aktoren können nicht unmittelbar ihre Position ändern. Daher kann es dazu kommen, dass die Markov Annahme verletzt wird, da Ergebnisse erst mehrere Zeitschritte später sichtbar werden.

4.3 Untermodelierte Modelle

Bei der Untermodellierung von Modellen werden nicht alle Parameter der realen Welt in einem Modell abgebildet. Durch die Nutzung von computergestützten Modellen können Robotern, anhand von Testdaten, Verhalten erlernen und eine optimale Policy finden. Der Vorteil einer Simulation ist klar zu erkennen. So können Kosten durch Wartung und Reparatur von Robotern vermieden werden und Task können beliebig oft wiederholt werden. Je nach Rechenleistung können Tasks in deutlicher höheren Geschwindigkeit erprobt werden.

In einer idealen Simulation könnte der Roboter das optimalste Verfahren bestimmen und das gleiche in der realen Welt anwenden. Dies entspricht jedoch nicht der Realität. Die Erstellung eines guten Modells ist meist schwer und erfordert eine große Anzahl an Daten. Durch kleine Fehler in dem simulierten Modell unterscheidet sich diese so stark, dass der gelernte Task in der realen Welt nicht angewendet werden kann.

Es kann zwischen stabilen Tasks und unstabilen Task unterschieden werden. Bei stabilen Task können die gewonnen Policies meist in die reale Welt übernommen werden, wohingegen unstabile Task meist überhaupt nicht übernommen werden können. Von stabilen Task wird gesprochen wenn es sich um ein selbststabilisierendes System handelt. Bei unstabilen Task haben kleine Veränderungen meist eine enorme Auswirkung auf das gesamte System.

4.4 Spezifikation der Ziele

In der Praxis ist es oft schwierig eine gute Reward-Funktion für eine Umgebung zu definieren. Der Agent ist jedoch auf ein gutes Belohnungssignal angewiesen, da er mit Hilfe diesem die optimale Policy finden muss. Liefert die Reward-Funktion immer das gleiche Ergebnis können keine Rückschlüsse getroffen werden ob eine Policy gut oder schlecht ist. Zusätzlich werden in der Robotik, im Gegensatz zu anderen Anwendungsbereiche von Reinforcement Learning, nicht nur am Ende eines Task eine Belohnung benötigt, sondern auch bei Zwischenschritten. Die Belohnungen würden ansonsten in einer realen Umgebung zeitlich zu weit auseinander liegen, sodass der Roboter keine Chance hätte in seiner Lebenszeit den Task erfolgreich abzuschließen.

Zusätzlich zu den Zwischenbelohnungen sollten Aktionen die die Lebensdauer des Roboters beeinträchtigen bestraft werden. Reinforcement Learning Algorithmen neigen zudem zur Ausnutzung von Grenzfällen der Reward-Funktion. Meist soll Robotern ein menschliches Verhalten, bei dem die benötigte Zeit und Risiken möglichst gering sind, gelernt werden. Um dies zu erreichen gibt es auch die Möglichkeit die Reward Funktion manuell zu spezifizieren. Die Reward Funktion wird anhand einer Demonstration eines menschlichen Experten rekonstruiert.

Kapitel 5

Fazit

Mit Hilfe von Reinforcement Learning und virtuellen Welten können Roboter Tasks deutlich schneller gelernt werden. Diese sogenannten virtuellen Klassenräumen befähigen Robotern Task in wenigen Tagen zu lerne, für die sie früher mehrere Wochen gebraucht hätten.

Trotz vieler Herausforderungen in der Robotik (siehe Kapitel 4) wie Mehrdimensionalität, untermodellierte Modelle, die reale Umgebung und schwierige Spezifikation der Ziele gelingt es immer wieder Fortschritte in diesem Forschungsgebiet zu erreichen.

Reinforcement Learning bietet auch der Industrie neue Möglichkeiten. Bisher sehr schwierige Task können mit Hilfe von Reinforcement Learning Algorithmen schneller den Robotern beigebracht werden. Zusätzlich können Aufgaben optimiert und verbessert werden. Die Programmierung neuer Fertigungsstraßen erforderte bisher viel Arbeitszeit und war mit hohen Kosten verbunden, jetzt können diese mit Reinforcement Learning Algorithmen geplant werden.

“It’s got the potential to really revolutionize what we can do in the robotics domain”[Bro], sagte ein Raia Hadsell ein Wissenschaftler der Firma Google DeepMind. Die Potenziale von Reinforcement Learning sind noch nicht ausgeschöpft, in den nächsten Jahren ist ein deutlicher Fortschritt insbesondere in der Robotik zu erwarten.

Literatur

[Bro] Mike Brown. *'Reinforcement Learning' Is Teaching Robots Faster Than Ever.* URL: https://www.inverse.com/article/21290-deep-reinforcement-learning-teaching-artificial-intelligence [siehe S. 11].

[Hei+07] Verena Heidrich-Meisner u. a. „Reinforcement learning in a nutshell.“ In: *ESANN*. Citeseer. 2007, S. 277–288 [siehe S. 6].

[KBP13] Jens Kober, J. Andrew Bagnell und Jan Peters. „Reinforcement learning in robotics: Asurvey“. In: *The International Journal of Robotics Research* [2013] [siehe S. 8].

[Shi] Nahum Shimkin. *Reinforcement Learning – Basic Algorithms*. URL: http://webee.technion.ac.il/people/shimkin/LCS11/ch4_RL1.pdf [siehe S. 5].

[SB98] Richard S Sutton und Andrew G Barto. *Reinforcement learning: An introduction*. Bd. 1. 1. MIT press Cambridge, 1998 [siehe S. 2].