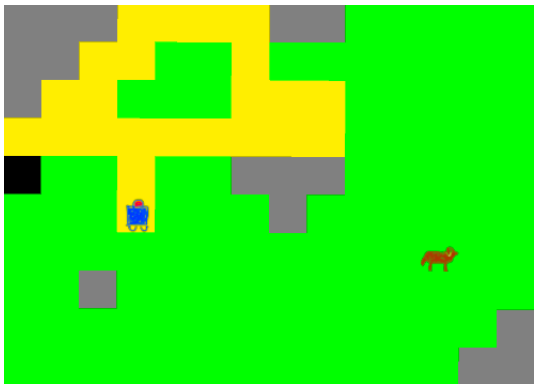
Gruppe 2: Smart Mower

Vorlesung Reinforcement Learning, WS 2015/16, Universität Augsburg

Moritz Einfalt Simon Stieber



Ziel dieses Projektes ist es, die Steuerung eines Rasenmähroboters in einer diskreten zweidimensionalen Gartenwelt mithilfe von Reinforcement Learning zu erlernen. Dazu werden der Q-Learning Algorithmus sowie verschiedene Erweiterungen von diesem verwendet. Die Ergebnisse zeigen, dass modellbasiertes Q-Learning im Vergleich die mit Abstand besten Ergebnisse liefert.

Das Problem

Die zu lösende Aufgabe besteht darin, eine möglichst effiziente Steuerung für einen autonomen Rasenmähroboter zu finden. Dieser agiert in einem Garten fester Größe, dessen genauer Aufbau anfänglich jedoch nicht bekannt ist. Dem Roboter stehen dabei nur Sensoren zu Verfügung, mit denen er seine lokale Umgebung wahrnehmen kann. Die Idee dieses Projekts ist es, das Erlernen einer solchen Steuerung als Reinforcement Learning Problem zu formulieren. Während Reinforcement Learning im Kontext der allgemeinen Robotik bereits intensiv untersucht wurde[1], ist den Autoren keine Betrachtung dieses konkreten Anwendungsfalls bekannt.

[1] http://www.ias.tu-darmstadt.de/uploads/Publications/Kober_IJRR_2013.pdf

Abbildung 1: Farbcodierung der Gartenzellen sowie der beweglichen Objekte.

Modellierung

Ausgangspunkt für den Rasenmähroboter ist dessen Umgebung – der Garten den es zu mähen gilt. Dieser wird modellhaft als diskretes zweidimensionales Raster fester Größe aus einzelnen Gartenzellen repräsentiert. Jede dieser Zellen ist gleich groß und kann verschiedene Zustände einnehmen:

- Hohes Gras
- Gemähtes Gras
- Hindernis (z.B. ein Stein)
- Basisstation des Mähroboters

Eine entsprechende Farbcodierung dieser Zustände, wie sie auch in der Simulation verwendet wird, ist in Abbildung 1 zu sehen. Der Mähroboter ist mit Sensoren ausgestattet, um den Zustand der umliegenden Gartenzellen wahrzunehmen.
Innerhalb des Gartens können sich verschiedene Objekte frei bewegen:

- Der Mähroboter
- Bewegliche Hindernisse (z.B. Hunde)

Diese werden in der Simulation wiederum gemäß Abbildung 1 dargestellt. Während die Steuerung des Roboters erlernt werden soll, bewegen sich die beweglichen Hindernisse zufällig durch die Gartenwelt.
Alle Bewegungen von Objekten im Garten erfolgen in diskreten Zeitschritten. Dies ist zusammen mit der diskreten Modellierung des Gartens den verwendeten Reinforcement Learning Algorithmen geschuldet. Diese erfordern zwingend ein Problem diskreter Natur mit endlicher Komplexität.
Um das Problem mithilfe von Reinforcement Learning zu lösen ist es notwendig, den Zustandsraum und die Aktionen des Roboters sowie die verwendete Belohnungsfunktion zu definieren. Dies wird im Nachfolgenden beschrieben.

Zustandsraum

Der Zustand des Roboter beschreibt das Wissen über sich selbst und seine Umgebung, auf Basis dessen er die Entscheidung über seine nächste Aktion trifft. Um die Größe des Zustandsraums übersichtlich zu halten wird der Zustand so modelliert, dass er nur die lokale und direkt wahrnehmbare Umgebung des Roboters beschreibt. Im Detail werden drei Zustandsvarianten untersucht:

Abbildung 2: Vergleich der Zustandsvarianten: Links enthält der Zustand des Roboters die Zustände der umgebenden vier Gartenzellen, rechts der umliegenden acht.

1. Zustand der umgebenden vier Gartenzellen
2. Zustand der umgebenden acht Gartenzellen
3. Zustand der umgebenden acht Gartenzellen + Koordinaten des Roboters

Die ersten beiden Varianten sind in Abbildung 2 dargestellt.
Die maximale Größe des Zustandsraums ist für Variante drei damit gegeben als $\{\text{Anzahl Zustände einer Gartenzelle}\}^8 * \{\text{Gartengröße}\}$.

Aktionen

Die möglichen Aktionen des Mähroboters beschränken sich auf Bewegungen in einer der vier Himmelsrichtungen. Eine Bewegung ist nur dann möglich, wenn die entsprechende Gartenzelle nicht durch ein Hindernis belegt ist.
Bewegt sich der Roboter auf eine Gartenzelle mit hohem Gras, so gilt diese Zelle als gemäht. Vom konkreten Mähvorgang dieser Zelle wird in der Simulation abstrahiert.

Belohnungsfunktion

Die Belohnungsfunktion zur Bewertung der Aktionswahl des Roboters ist sehr einfach definiert. Der Roboter erhält nach jeder Aktion eine Belohnung von

- $+10$, falls er sich auf ein Feld mit hohem Gras bewegt hat,
- -1, sonst.

Algorithmen

Ziel der verwendeten Algorithmen ist es immer, basierend auf dem aktuellen Zustand des Mähroboters die optimale Aktion vorauszusagen. Als Ausgangspunkt wird der Q-Learning Algorithmus gewählt, welcher sich durch seine simple Implementierung auszeichnet.
Als Erweiterung davon werden zudem Eligibility Traces verwendet, um auch ganze Bewegungsabläufe des Roboters erfassen zu können.

Abbildung 3: Rechts: Die reale Gartenwelt.
Links: Das unvollständige Umgebungsmodell des Roboters. Nach jeder Bewegung werden weitere zufällige Bewegungen im Modell simuliert und darauf gelernt.

Als dritte Variante wird zudem noch eine modellbasierte Erweiterung von Q-Learning verwendet, welche auf dem DynaQ Algorithmus basiert. Dabei erstellt und aktualisiert der Roboter nach und nach ein internes Modell des Gartens, auf welchem er dann zusätzliche Aktionswahlen simulieren und darauf lernen kann. Abbildung 3 zeigt den Zusammenhang zwischen der echten Gartenwelt und dem internen Modell.

Messbarkeit

Als Maß für die Güte der Robotersteuerung wird die Anzahl an Zeitschritten gemessen, bis der Roboter den gesamten Garten gemäht hat. Dies wird über mehrere Episoden wiederholt.

Umsetzung und Ergebnis

Die Simulation

Die Simulation wird im Rahmen eines Unity-Projekt realisiert, um auch eine geeignete graphische Ausgabe zu realisieren. Dabei werden die Simulationslogik sowie die verwendeten Lernalgorithmen jedoch so gekapselt, dass sie auch außerhalb von Unity verwendet werden können. Abbildung 4 zeigt die graphische Oberfläche der Simulation. Dabei wird die bereits bekannte Farbcodierung aus Abbildung 1 verwendet.

Experimente: Vergleich der Algorithmen

Alle drei Varianten des Algorithmus werden über 50 Episoden trainiert. Dabei werden jeweils die Zeitschritte gemessen, bis der gesamte Rasen gemäht ist. Für alle Varian-

Abbildung 4: Screenshot der Unity Simulation.

ten wird durch eine Parameter-Optimierung eine optimale Parameterkombination ermittelt. Die exakten Parameter sind im Appendix aufgelistet. Zusätzlich wird als Vergleichswert eine Robotersteuerung mit zufälliger Aktionswahl verwendet. Als Gartenwelt wird die Welt aus Abbildung 4 mit 123 zu mähenden Graszellen verwendet. Abbildung 5 zeigt die Ergebnisse der Experimente.

Als oberer Vergleichswert dient das Ergebnis der zufälligen Aktionswahl bei durchschnittlich ~1750 Schritten pro Episode.
Mit reinem Q-Learning können im Mittel ~1200 erreicht werden. Dies ist bei 123 zu mähenden Graszellen jedoch immer noch in etwa das Zehnfache des theoretischen Optimums. Auffallend ist, dass sich der erreichbare Lernerfolg schon nach wenigen Episoden einstellt und dann nicht mehr verbessert. Zudem sind immer wieder einzelne Episoden zu erkennen, in denen sehr große „Ausreißer“-Werte beobachtet werden können. Diese wirken sich entsprechend negativ auf den Mittelwert über alle 50 Episoden aus.
Auch die Erweiterung mit Eligibility Traces bringt keine Verbesserung mit sich – im Gegenteil. Der Mittelwert der benötigten Zeitschritte liegt hier sogar leicht über dem von reinem Q-Learning. Der Verlauf über die Zeit ist bei beiden Algorithmen sehr ähnlich.
Vollkommen anders ist das Ergebnis des modellbasierten Q-Learnings. Hier werden über alle Episoden hinweg im Mittel nur ~180 Zeitschritte benötigt, was einem Optimum bereits sehr nahe kommt. Auch einzelne Episoden mit Extremwerten wie bei den vorigen Algorithmen sind hier nicht zu beobachten. Doch auch hier stellt sich der erreichbare Lernerfolg bereits nach sehr wenigen Episoden ein.
Die für die Experimente verwendete Gartenwelt enthält mit nur einem sich bewegenden Hund sehr wenig Dynamik. Ob die verschiedenen Algorithmen in einer Welt

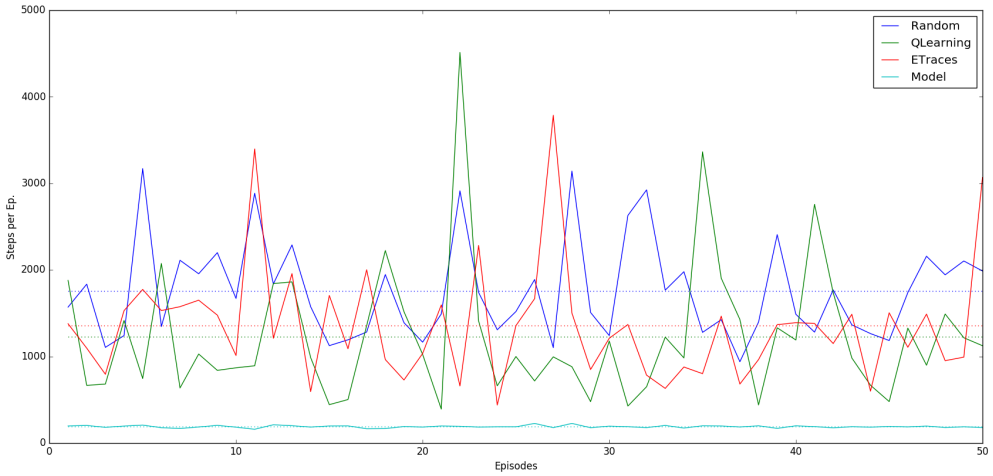


Abbildung 5: Ergebnisse der verschiedenen Algorithmen über 50 Episoden: Zufällige Aktionswahl, Q-Learning, Q-Learning mit Eligibility Traces, Modellbasiertes Q-Learning. Der Mittelwert über alle Episoden hinweg ist gepunktet angetragen.

mit mehr Dynamik ähnlich abschneiden, muss in weiteren Experimenten untersucht werden.

Appendix: Optimale Parameterkombinationen

Nachfolgend die Auflistung der optimalen Parameterkombinationen, welche für die verschiedenen Algorithmen ermittelt wurden. Die Benennung der Parameter entspricht der üblichen Benennung die in der Literatur zu Reinforcement Learning zu finden ist.

Tabelle 1: Optimale Parameter für reines Q-Learning.

Parameter	Wert	Beschreibung
α	0.6	Lernrate
γ	0.8	Diskont-Wert
ε	0.5	Greediness

Tabelle 2: Optimale Parameter für Q-Learning mit Eligibility Traces.

Parameter	Wert	Beschreibung
α	0.7	Lernrate
γ	0.8	Diskont-Wert
ε	0.5	Greediness
λ	0.1	Trace-Länge

Tabelle 3: Optimale Parameter für modellbasiertes Q-Learning.

Parameter	Wert	Beschreibung
α	0.6	Lernrate
γ	0.9	Diskont-Wert
ε	0.9	Greediness
N	5000	Simulierte Schritte pro Zeitschritt