Technische Hochschule
Brandenburg
University of
Applied Sciences
Fachbereich
Informatik und Medien

Kombination von Imitation Learning und Reinforcement Learning zur Bewegungssteuerung

Darya Martyniuk

Masterarbeit • Studiengang Informatik • Fachbereich Informatik und Medien • 22.01.2020

Zielsetzung

Ziel der Arbeit ist die Konzeption, Entwicklung und Evaluierung einer Softwarekomponente, die es einem NAO-Roboter ermöglicht, die Bewegungssteuerung für das Spiel Ball-in-a-Cup ohne der Verwendung einer Simulation zu erlernen. Der Schwerpunkt liegt auf der Auswahl und Umsetzung eines Lernalgorithmus.

Szenario: Ball-in-a-Cup

Die Idee des Spiels ist, einen Ball, der an einer Schnur aus einem Becher hängt, in Schwingungen zu bringen und mit dem Becher zu fangen. Dafür muss der lernende Roboter (blauer Roboter, Abb. 1) seinen rechten Arm, der sechs Freiheitsgrade besitzt, einsetzen. Zur Vereinfachung der Aufgabe wird die Anzahl der beweglichen Armgelenke auf drei reduziert. Zur Ermittlung der Ballposition während des Spiels wird ein zweiter NAO-Roboter (roter Roboter, Abb. 1) verwendet.

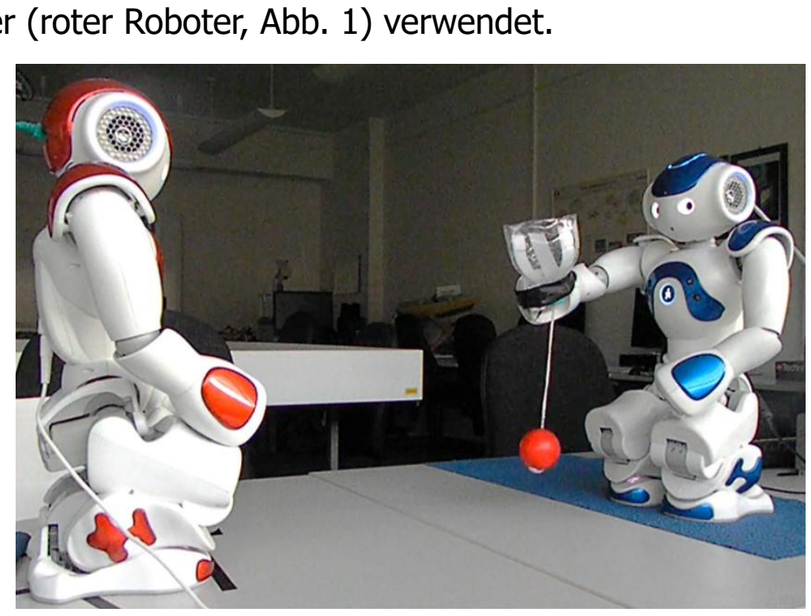

Abb. 1: Versuchsaufbau für das Ball-in-a-Cup Spiel

Methodik

Das Lernproblem wird als Markov-Entscheidungsprozess (MDP) mit einem kontinuierlichen Zustandsraum und einem kontinuierlichen Aktionsraum formalisiert. Die Grundlage für das Lernen stellen kinästhetische Demonstrationen eines Experten sowie eigene Erfahrung des Agenten, die er durch die Interaktion mit der Umgebung sammelt, dar. Die Lernkomponente basiert auf dem Algorithmus *Deterministic Policy Gradient from Demonstration(DDPGfD) [1],* der in der Arbeit auf das Clipped-Double-Q-Learning und die Glättung der Target-Policy (inspiriert vom Algorithmus *Twin Delayed Policy Gradient (TD3) [2]*) erweitert wird. Der umgesetzte Lernalgorithmus kann in der Klasse der modellfreien Actor-Critic-Algorithmen mit Demonstrationen eingeordnet werden. Für die Approximation der optimalen Wertefunktion sowie der optimalen Strategie des Roboters werden *künstliche neuronale Netze*, Deep-Feedforward-Netze, eingesetzt. Die Exploration der Umgebung findet durch das Verrauschen der aktuellen Policy des NAO-Roboters mit Gaußschem Rauschen statt. Das Gedächtnis des Roboters bildet ein Replay-Puffer einer begrenzten Größe, in dem die Demonstrationen des Experten sowie eigene Erfahrung des Roboters gespeichert werden. Beim Erreichen der maximalen Größe des Replay-Puffers wird nur die alte Erfahrung des Roboters mit der neuen Erfahrung überschrieben.

Implementierung

Die Implementierung des Lernalgorithmus erfolgte in der Programmiersprache Python. Zur Steuerung des NAO-Roboters wurde das NAOqi-Framework verwendet und zur Erstellung und Aktualisierung der neuronalen Netze wurde das PyTorch-Framework eingesetzt. Die komplexen Berechnungen fanden auf der Grafikkarte NVIDIA Titan RTX statt.

Evaluation

Um den Lernalgorithmus zu validieren sowie die Intuition für die Bestimmung der Hyperparameter für das Lernszenario zu bekommen, wurde der Algorithmus zuerst in zwei virtuellen Umgebungen von OpenAI [3] („Pendulum-v0" und „LunarLander-Continuous-v2") mit unterschiedlichen Parameterkonfigurationen evaluiert. Dabei standen die Performanz des Agenten und die Reproduzierbarkeit der Ergebnisse im Vordergrund.
Im finalen Trainingslauf vom Spiel Ball-in-a-Cup wurden 200 Episoden ausgeführt. Das entsprach eine Interaktion mit der Umgebung von 89 Minuten. Aufgrund der Überhitzung der Robotergelenke, dauerte das gesamte Training jedoch länger. Der Roboter verfügte über 100 Demonstrationen. Die Abbildung 2 stellt den Verlauf der erlangten Gesamtbelohnung pro Episode während des Lernens dar. Den ersten Gewinn konnte der Roboter bereits nach 19 Episoden erzielen.

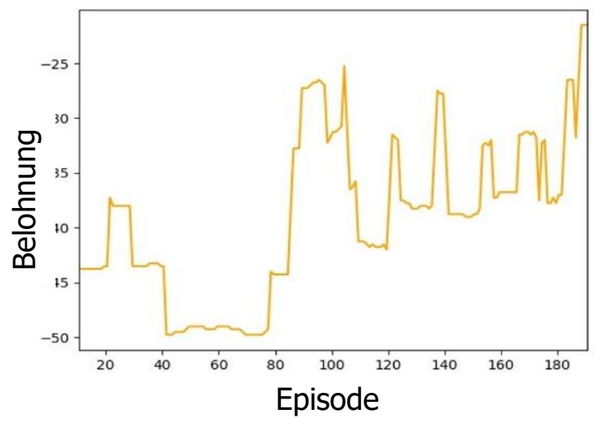

Abb. 2: Verlauf der Gesamtbelohnung über 200 Episoden im Spiel Ball-in-a-Cup. Die Kurve ist in einer geglätteten Form dargestellt

Fazit

Die Ergebnisse zeigen, dass der umgesetzte Algorithmus ein effizientes Lernen ermöglicht. Vortrainiert mit Demonstrationen, fängt der Roboter die Interaktion mit der Umgebung mit einer suboptimalen Strategie an, die er im Laufe des Trainings verbessert. Die Leistung des Algorithmus ist jedoch stark von der Konfiguration der Hyperparameter sowie der Anzahl der Demonstrationen abhängig. In zukünftigen Arbeiten soll daher für das Spiel eine Simulation erstellt werden, in der die Hyperparameter und mögliche Verbesserungen des Lernverfahrens vor dem Training mit dem realen Roboter evaluiert werden können.

Quellen

[1] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In Y. Bengio & Y. LeCun (Eds.), *ICLR*. *https:// arxiv.org/ abs/ 1509.02971*. Zugriff am 21.06.2019.
[2] Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *CoRR*, abs/1802.09477. *https: // arxiv.org/ pdf/ 1802.09477.pdf* . Zugriff am 13.10.2019.
[3] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *ArXiv*, abs/1606.01540. *https:// arxiv.org/ abs/ 1606.01540* . Zugriff am 26.11.2019.

Betreuung Dipl.-Inform. Ingo Boersch • Technische Hochschule Brandenburg • Prof. Dr.-Ing. Jochen Heinsohn • Technische Hochschule Brandenburg