

Verstärkungslernen mit realen Robotern*

Alexander Gloye[†]

10. November 2003

1 Einleitung

RoboCup ist eine jährlich stattfindende Veranstaltung für fußballspielende Roboter. Dessen Ziel ist die Förderung der Künstlichen Intelligenz und Robotik durch Wettbewerbe und Informationsaustausch. Die FU Berlin beteiligt sich mit ihren FU-Fighters seit 1998 in der Small-Size Liga und seit 2001 auch in der Mid-Size Liga. Die Software zur Steuerung der Roboter ist ein komplexes System, bestehend aus vielen Komponenten, wie zum Beispiel Bildverarbeitung, Verhaltenssteuerung, Kommunikation und Mikrocontroller-Programmierung. Lernverfahren könnten an mehreren Stellen im System die aufwändige Programmierung vereinfachen und die Gesamtleistung optimieren.

Im folgenden werden drei Ansätze vorgestellt, die benutzt wurden, um reale Roboter lernen zu lassen. In Kapitel 3 geht es um Lernverfahren, die von anderen RoboCup Teams eingesetzt werden. Kapitel 4 beschreibt eingesetzte und geplante Anwendungen von Lernmethoden bei den FU-Fighters.

2 Ansätze

Simulation

Bei der Simulation geht es in erster Linie um die Beschleunigung der Zeit, bei den anderen Verfahren um Problemvereinfachung durch Vorwissen. Das wohl offensichtlichste und auch klassische Verfahren um die Lehrzeit realer Systeme zu minimieren ist sicherlich die Simulation. Um so genauer die Simulation ist, um so kürzer ist auch die anschließende Feinabstimmung

*Beitrag zum Doktorandenworkshop des Instituts für Informatik der FU 24.-25.10.04

[†]gloye@inf.fu-berlin.de

des physikalischen Agenten. So wurde zum Beispiel ein Khepera Roboter trainiert, in einem Labyrinth umherzufahren, ohne an Wänden anzustoßen [Janusz]. Die Welt des Agenten bestand aus den Daten von acht Abstandssensoren. Der Agent konnte aus 5 Aktionen (langsam links, schnell links, vorwärts, langsam rechts und schnell rechts) wählen. Eine Episode war immer 48 Zeitschritte lang. Wenn der Roboter in dieser Zeit kein Hindernis berührte, gab es eine Belohnung. Nach 50000 Trainingsläufen wurde das Gelernte dann auf den realen Roboter übertragen. Die Erfolgsquote sank von 95% bei der Simulation auf 90% bei dem realen Roboter.

Modularisierung

Ein Beispiel wurde von [Kalmár] vorgestellt. Die Aufgabe für einen Khepera Roboter bestand darin, einen Ball zu suchen, diesen mit einem Greifarm aufzunehmen, zu einem Pfahl zu fahren und den Pfahl mit dem Ball zu berühren. Dem Roboter wurde die Aufgabe jedoch so gestellt, dass er aus einer Menge von Verhalten (die der oberen Vorgehensweise entsprechen) ein Verhalten auswählen kann und nur keine negative Belohnung bekam wenn er mit dem Ball den Pfahl berührte.

Für die Aufgabenstellung war die Aufteilung in die Module offensichtlich. Ausserdem sind die Teilaufgaben nicht abhängig voneinander. [Uchibe] hat ein Problem vorgestellt, in dem die Teilaufgaben stark interferieren: Die Aufgabe bestand aus den zwei Teilaufgaben „schieße ein Tor“ und „weiche dem Torwart aus“. Die Teilaufgaben wurden getrennt voneinander trainiert. Das Problem besteht jedoch darin, dass beim Zusammenfügen die beiden Verhalten teilweise gegensätzliche Ziele haben, zum Beispiel, wenn der Ball nahe am Torwart liegt. Es gibt unterschiedliche Ansätze, um dieses Dilemma zu lösen, auf die hier nicht genauer eingegangen wird [Asada94].

Learning from Easy Missions

Diese Formulierung wurde von [Asada96] eingeführt. Die Idee ist es, den Agenten zu Beginn des Lernverfahrens nur mit einfachen Situationen nahe am Ziel zu konfrontieren und dem Agenten schrittweise schwierigere Aufgaben zuzuweisen. Man nutzt das Vorwissen also nicht, um das Problem direkt zu vereinfachen, sondern indirekt über eine Variation der Aufgabenstellung. Das Verfahren ist vergleichbar mit dem Schritt von Q-Learning, in dem man nur die Bewertung des nächsten Schritts berücksichtigt, zu TD(λ)-Learning, in dem man viele Schritte in die Zukunft schaut.

3 Lernen beim RoboCup

Aufgrund des hohen Lernaufwands gab es bisher vorwiegend in der Simulationsliga Implementierungen. Ein Beispiel ist das 3 vs. 2 Keepaway [Stone]. Dabei spielen 3 Angreifer gegen zwei Verteidiger auf einem quadratischen Spielfeld. Die Angreifer sind in Ballbesitz und deren Aufgabe ist es, den Ball möglichst lange zu behalten. Wenn sie den Ball an die Verteidiger verlieren oder der Ball außerhalb des Spielfelds ist, ist das Spiel beendet. Die einzige Strategie der Verteidiger besteht darin, zum Ball zu laufen. Sie werden nicht trainiert. Die Angreifer können aus den Aktionen „Ball halten“, „Ball passen“, „Gehe zum Ball“ und „freistellen“ auswählen. Es ist also eine Modularisierung vorhanden. Die Agenten lernten mit Q -Learning nach 20000 Episoden, den Ball durchschnittlich 14 Sekunden zu behalten.

Von [Kleiner] wurde ein Experiment in die Mid-Size Liga umgesetzt. Die ersten Versuche wurden auch hier mit einem Simulator durchgeführt und später mit einem realen Roboter validiert. Das Ziel war es, unterschiedliche Eigenschaften der Roboter zu kompensieren. So waren die Roboter zwar alle gleich aufgebaut, aber die Schussstärke war etwas unterschiedlich. Es sollten aber nicht die Verhalten der Roboter von Hand optimiert werden, sondern die Anpassung der Verhalten sollte gelernt werden. Die Grundverhalten waren bei allen Robotern gleich, aber die Aktivierung der Verhalten wurde mit Hilfe des Q -Learning Algorithmus gelernt. Dadurch haben Roboter mit einem starken Schuss früher geschossen und Roboter mit einem schwachen Schuss etwas später. Die Anzahl der erfolgreichen Torschüsse stieg dadurch stärker an, als durch handoptimierte Parameteranpassung.

4 Lernen bei den FU-Fighters

Bisherige Umsetzung

In der RoboCup Small-Size Liga erreichen die Roboter eine Geschwindigkeit von bis zu 2m/s. Das Regelungssystem zur Steuerung der Roboter hat jedoch eine Verzögerung von 130ms. Bis ein Roboter auf eine Situation reagieren kann weicht seine Position bis zu 26cm von der erwarteten Position ab. Eine vorausschauende Fahrweise ist somit wichtig. Diese wird durch eine Vorhersage um die Länge der Verzögerung erreicht. Die Vorhersage geschieht entweder mit Hilfe eines neuronalen Netzes [Behnke] oder eines linearen Modells [Gloye]. Die Eingaben und Ausgaben sind für beide Modelle gleich: Die Eingabe besteht aus den Zustandparametern¹ (Position-, Orientierung- und Sollgeschwindigkeitsvektor) des Roboters bezüglich der aktuellen Position

¹Bei der Vorhersage fremder Roboter entfällt bei der Eingabe die Sollgeschwindigkeit.

über die sechs letzten Zeitschritte. Die Ausgabe besteht aus den erwarteten relativen Position und Orientierung nach 130ms bezüglich der aktuellen Position und Orientierung. Die Vorhersage hat einen durchschnittlichen quadratischen Fehler von 1,32cm pro Dimension.

Das Verhalten „Fahren zum Ort“ ist ein elementares Verhalten in der Verhaltenssteuerung der FU-Fighters. Das Verhalten berechnet anhand von gewünschter Maximalgeschwindigkeit, Zielposition, Zielorientierung, Zielgeschwindigkeit, momentaner Position, momentaner Geschwindigkeit und momentaner Orientierung den gewünschten Geschwindigkeitsvektor für den Roboter. Das Ziel ist es, in möglichst kurzer Zeit mit einer akzeptablen Positionstoleranz an der Zielposition anzukommen. Die Hardware des Roboters ist jedoch Beschränkungen unterworfen. So kann er nur mit einer gewissen Trägheit seine Richtung ändern und er darf nur mit einer begrenzten Kraft bremsen, um nicht umzukippen. Es müssen also eine Menge Parameter eingestellt werden, um den Roboter optimal zu steuern. Natürlich unterscheiden sich die Roboter auch geringfügig, sodass jeder Roboter einzeln konfiguriert werden sollte. Diese Einstellungen von Parametern wurde deshalb mit einer Monte-Carlo Methode automatisiert. Der Bremsweg von der Maximalgeschwindigkeit bis zum Stillstand reduzierte sich dadurch von 40cm auf unter 35cm.

Geplante Vorhaben

Es gibt mehrere Verhalten, bei denen das Lernen in Zukunft angewendet werden soll. Wenn der Ball nicht direkt auf den Roboter zurollt, muss die optimale Abfangposition berechnet werden. Die optimale Abfangposition wird von unzähligen Parametern des Systems beeinflusst, die oft auch mit einem großen Fehler behaftet sind, wie zum Beispiel die Ballgeschwindigkeit. Je nach geschätzter Ballgeschwindigkeit kann es zu Situationen kommen, wo entweder an der Abfangposition gewartet werden muss oder der Ball verpasst wird. Um solche Fehler minimal zu halten, kann man das Verhalten „Ball halten“ automatisch lernen lassen und so das Abfangen optimieren.

Ein anderes aktuelles Problem ist Dribbeln — das kontrollierte Ballführen nah am Roboter. Tests zeigen, dass eine schnelle und sichere Ballführung möglich ist. Die Anpassung der Parameter, zum Beispiel die Abhängigkeit zwischen Dreh- und Fahrgeschwindigkeit, ist jedoch nicht trivial. Lernen kann hier eingesetzt werden, um diese Anpassung automatisch durchzuführen. Dies benötigt aber ein explizites Modell, das die physikalischen Abhängigkeiten eventuell nicht hinreichend widerspiegelt. Deshalb soll auch versucht werden ein Modell zu lernen, indem man aufgezeichnete Daten aus manuellen Tests für das Training benutzt.

Ein weiteres geplantes Vorhaben ist die Optimierung und/oder Ersetzung der PID-Regler auf der Roboterhardware. Einerseits soll explizites Vorwissen benutzt werden, um nur die Parameter der PID-Regler zu optimieren, andererseits sollen die Regler ersetzt bzw. erweitert werden, um das systemimmanente Überschwingen der Regler zu eliminieren. Hierfür gibt es vergleichbare Projekte. In [Riedmiller] wurde ein PID-Regler für ein Saugrohrdrucksystem, in dem die Stellung einer Drosselklappen gesteuert wurde durch ein neuronales Netz optimiert. Die besten Ergebnisse wurden erzielt, indem dem vorhandenen Regler ein neuronaler Regler vorgeschaltet wurde.

Literatur

- [Asada94] Minoru Asada, Eiji Uchibe, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda: *Coordination Of Multiple Behaviors Acquired By Vision-Based Reinforcement Learning*, in Proc. of IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems, Munich, Germany, 1994.
- [Asada96] Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda: *Purposive Behavior Aquisition for a Real Robot by Vision-Based Reinforcement Learning*, in Machine Learning, vol. 23, pp. 279-303, 1996.
- [Behnke] Sven Behnke, Anna Egorova, Alexander Gloye, Raúl Rojas, and Mark Simon: *Predicting Away the Delay*, in RoboCup2003 — Proc. of the Int. Symposium, Padova, Italy, 2003.
- [Gloye] Alexander Gloye, Mark Simon, Anna Egorova, Fabian Wiesel, Oliver Tenchio, Michael Schreiber, Sven Behnke, and Raúl Rojas: *Predicting away robot control latency* Technical Report B-08-03, Institut für Informatik, Freie Universität Berlin, 2003.
- [Janusz] Barbara Janusz, and Marin Riedmiller: *Self-learning neural control of a mobile robot*, in Proc. of the IEEE ICNN 95, Perth, Australia, 1995.
- [Kalmár] Zsolt Kalmár, Csaba Szepesvári, and András Lőrincz: *Module Based Reinforcement Learning for a Real Robot*, in Machine Learning, vol. 31, pp. 55-85, 1998.
- [Kleiner] Alexander Kleiner, Markus Dietl, and Bernhard Nebel: *Towards a Life-Long Learning Soccer Agent*, in RoboCup 2002, G. A. Kaminka, P. U. Lima, R. Rojas (Eds.), LNAI 2752, pp. 126-134, Springer Verlag, 2003.
- [Riedmiller] Martin Riedmiller, Ralf Schoknecht: *Einsatzmöglichkeiten selbständig lernender neuronaler Regler im Automobilbereich*, in VDI Berichte 1381, pp 259-275, Berlin, 1998.
- [Stone] Peter Stone, Richard S. Sutton, and Satinder Singh: *Reinforcement Learning for 3 vs. 2 Keepaway*, in RoboCup-2000: Robot Soccer World Cup IV, P. Stone, T. Balch, and G. Kraetschmar, Eds., Springer Verlag, 2001.
- [Uchibe] Eiji Uchibe, Minoru Asada, and Koh Hosoda: *Behavior Coordination for a Mobile Robot Using Modular Reinforcement Learning*, in Proc. of. IEEE/RSJ Int. Conf. on Intelligent Robot and Systems, pp. 1329-1336, 1996.