Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 1 WiSe 2010

Jan Busch

Modellbasiertes Reinforcement Learning im Kontext des FAUST-Projektes

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Kurzzusammenfassung

Diese Ausarbeitung beschäftigt sich mit Ansätzen zur Modellbildung im Bereich des Reinforcement Learning[1]. Beginnend mit einer kurzen Einführung in das Themengebiet und den Kontext der Arbeit, ist es das Ziel, Probleme und Möglichkeiten von modellgestütztem Reinforcement Learning aufzuzeigen. Desweiteren wird diskutiert, ob und inwiefern solche Verfahren zum Erfolg des FAUST-Projektes beitragen können und welche Vorteile sie gegenüber konventionellen Verfahren haben. Die Risiken werden beleuchtet und ein Ausblick über zukünftige Arbeiten in diesem Bereich gegeben.

1 Einleitung

1.1 Forschungsprojekt FAUST

Das FAUST-Projekt (FAUST (2011)) aus dem Department für Informatik der Hochschule für Angewandte Wissenschaften Hamburg entwickelt Technologien für **F**ahrerassitenz- und **Au**tonome **Syst**eme auf verschiedenen selbstentwickelten Fahrzeugplattformen. Die Schwerpunkte der verschiedenen Projekte liegen hierbei auf:

- Sensorik, Telemetrie und Bildverarbeitung
- Echtzeitsystemen und Bussystemen
- Software- und Hardwarearchitekturen
- Algorithmik und Steuerung

In vielen aktuellen Fahrzeugen finden sich bereits Assistenzsysteme in Form von Einparkhilfen oder Abstandsreglern. Die große Anzahl dieser Helfer wird in Zukunft noch weiter steigen, um den Fahrer optimal zu unterstützen und dessen Sicherheit zu steigern.

Verschiedene Fahrzeuge des FAUST-Projektes nehmen jährlich am Carolo-Cup (Carolo-Cup (2011)) der technischen Universität Braunschweig teil. Bei diesem Hochschulwettbewerb müssen die Fahrzeuge selbstständig einen Parcours mit Hindernissen bewältigen, automatisch einparken und weitere Probleme lösen. Ziel ist es, die verschiedenen Disziplinen möglichst schnell und fehlerfrei zu absolvieren. Die Jury setzt sich aus Experten aus Wissenschaft und Industrie zusammen und bewertet die Ergebnisse sowie die Konzepte hinter den Ergebnissen. Die beiden Teams der HAW arbeiten stetig an der Verbesserung ihrer Fahrzeuge und deren Software.

1.2 Simulationsumgebung TORCS

TORCS steht für „**T**he **O**pen **R**acing **C**ar **S**imulator“ und ist eine Simulationsumgebung für Rennwagen. Als Multiplatform-Software steht sie für Linux (x86, AMD64 and PPC), FreeBSD, MacOSX und Windows zur Verfügung. Der Source-Code ist unter der „General Public License“[2] (GPL) lizensiert und wird häufig als Plattform für Forschungszwecke verwendet. Darüber hinaus bietet sie viele Möglichkeiten:

- Programmierung der künstlichen Intelligenz der Fahrzeuge
- Design eigener Fahrzeuge, d.h. Aussehen und Fahrverhalten
- Erstellung eigener Strecken
- Bereitstellung von Informationen über die Umgebung der Fahrzeuge und über das einzelne Fahrzeug selbst
  - z.B. Fahrbahnhaftung, Geschwindigkeit, Beschleunigung und Lenkwinkel

Abbildung 1: Screenshot aus TORCS

Abbildung 1 zeigt einen Screenshot aus der Rennsimulation von TORCS. Auf verschiedenen, ggf. selbst erstellten Strecken, können Verhalten der programmierten Fahrzeuge unter unterschiedlichen Bedingungen beobachtet werden.

[2] siehe http://www.gnu.org/licenses/licenses.html

1.3 Motivation: Wozu modellbasiertes Lernen?

Die Steuerung der Fahrzeuge, d.h. Einstellung des Lenkwinkels, Geschwindigkeit und das gesamte Verhalten, ist zur Zeit „hard-coded“[3]. Die Umgebung der Fahrzeuge und die damit verbundenen Anforderungen sind jedoch sehr variabel. Auch die Fahrzeuge selbst sind bezüglich des Lenkwinkels und anderer kleinerer Abweichungen in der Steuerung nicht perfekt einstellbar. Das führt dazu, dass sich dieses System nicht ohne Weiteres um den Arbeitspunkt linearisieren lässt und es daher sehr kompliziert ist, einen Regler für dieses System zu konstruieren.

Eine gute Möglichkeit, ein System optimal zu steuern, ohne aber das genaue Verhalten für dieses System bestimmen zu können, sind maschinelle Lernverfahren. Eines dieser maschinellen Lernverfahren, das seit langem genutzt wird, ist Reinforcement Learning. Reinforcement Learning kommt mit sehr geringen Informationen aus und führt trotzdem zu einer optimalen Strategie. Hierauf gehe ich in Kapitel 2 genauer ein. Der Bereich des Reinforcement Learning umschreibt ein großes Spektrum an Methoden und Verfahren, arbeitet aber selten mit Modellen der Umwelt und wenn, meist auf Basis vorgegebener Modelle. Dies bedeutet, dass das Lernen eines Agenten auf einer sehr niedrigen Ebene angesiedelt ist und dieser keine Erfahrungen von bereits erhaltenen Belohnungen auf zukünftige Zustände bzw. Aktionen übertragen kann. Er ist nicht in der Lage, Vorhersagen über die zu erwartende Belohnung und den möglichen Folgezustand zu treffen.

Für Probleme, wie beispielsweise dem Lernen der optimalen Geschwindigkeit im FAUST-Kontext könnte es jedoch von Vorteil sein, nicht alle möglichen Zustände erst durchlaufen zu müssen. Auf Basis bereits bekannter Zustände, können Schlüsse auf noch nicht Besuchte gezogen werden. Desweiteren beschleunigt die Verwendung eines Modells zur Entscheidungsfindung deutlich das Lernen (Amato und Shani (2010)).

Die Geschwindigkeit ist eines der Bewertungskriterien im Carolo-Cup, jedoch dürfen keine Fehler bei der Passage des Parcours auftreten. Daher wäre es sinnvoll, eine Möglichkeit zu finden, die Steuerung der optimalen Geschwindigkeit automatisch erlernen zu lassen. Die Geschwindigkeit sollte hierbei so hoch wie möglich sein, aber das Fahrzeug sollte nicht Gefahr laufen, von der Fahrbahn abzukommen.

1.4 Ziel dieser Arbeit

Das Ziel dieser Arbeit ist einen geeigneten Lösungsansatz zu finden, der hilft, modellbildende Methoden aus dem Reinforcement Learning zu finden, die sich auf Probleme aus dem

[3] engl. für „hart-kodiert“: Bedeutet, dass zur Laufzeit keine Veränderungen am Verhalten der Programmierung möglich sind.

FAUST-Projekt anwenden lassen. Die Ergebnisse können ggf. so generalisiert werden, dass sie sich ohne Weiteres auf andere Problemfelder und Anwendungsgebiete übertragen lassen. Auf diese Weise kann sowohl ein Beitrag zum FAUST-Projekt als auch zur wissenschaftlichen Weiterentwicklung von Lernalgorithmen geleistet werden.

1.5 Übersicht über diese Arbeit

Diese Arbeit ist in 5 Kapitel unterteilt.

Das erste Kapitel soll einen Überblick über das FAUST-Projekt, die Simulationsumgebung TORCS und meine Motivation verschaffen. Desweiteren wird das Ziel definiert und eine kleine Übersicht über diese Arbeit gegeben.

In Kapitel 2 wird auf Reinforcement Learning im Allgemeinen eingegangen, um dem Leser die Funktionsweise zu verdeutlichen und die Basis für tiefergehende Informationen diesbezüglich zu bilden.

Kapitel 3 analysiert und diskutiert die Architektur von Dyna. Fragestellung ist, ob sich Dyna als Basis für weiterführende Arbeiten eignet bzw. ob es bereits auf Dyna aufbauende Arbeiten gibt, die hilfreich sein könnten.

Der von mir verfolgte Lösungsansatz wird in Kapitel 4 vorgestellt und erläutert. Anhand dessen werden Vor- und Nachteile diskutiert und die Risiken aufzeigt.

Abschließend werden in Kapitel 5 die Ergebnisse dieser Arbeit zusammengefasst und es wird ein Ausblick auf den weiteren Verlauf dieses Projektes gegeben.

2 Einführung in Reinforcement Learning

> „*A mobile robot decides whether it should enter a new room in search of more trash to collect or start trying to find its way back to its battery recharging station. It makes its decision based on how quickly and easily it has been able to find the recharger in the past.*“ Sutton und Barto (1998)

Reinforcement Learning dient als Überbegriff für eine vielzahl verschiedener Methoden im Bereich des maschinellen Lernens. Seit über zwanzig Jahren wird es in den Bereichen der künstlichen Intelligenz eingesetzt und weiterentwickelt.

Hierbei wird ein relativ simpler Lernansatz verfolgt (siehe Abbildung 2): Ein Agent nimmt seine Umwelt über eine Sensorik in diskreten Zeitschritten t wahr. Aus diesen Informationen ergibt

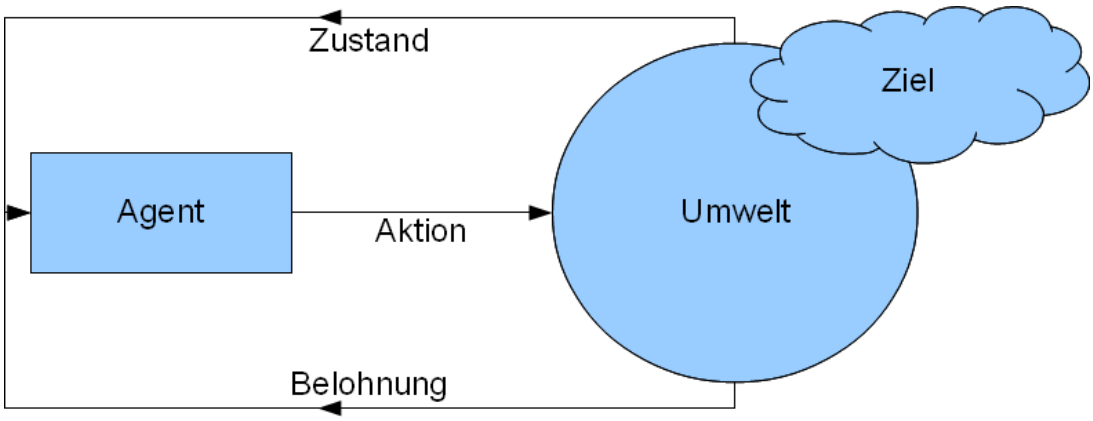

Abbildung 2: Reinforcement Learning

sich der Zustand s_t des Agenten. Auf Basis einer Strategie p wählt er unter der Berücksichtigung von s_t eine Aktion a_t und führt diese aus. Abhängig von den Folgen der gewählten Aktion erhält der Agent eine Belohnung r_t über eine Belohnungsfunktion, die jedem Zustand oder auch einem Zustands-Aktions-Paar eine Belohnung in Abhängigkeit vom Ziel zuordnet. Diese unmittelbaren Belohnungen nehmen Einfluss auf die Wertefunktion, welche die Langzeitbelohnungen für jeden Zustand (Zustands-Aktions-Paar) beschreibt und Basis für die Entscheidungen des Agenten sind bzw. Einfluss auf die Strategie nehmen. Die gewählte Aktion führt desweiteren dazu, dass sich die Umwelt verändert und in einen Folgezustand s_{t+1} übergeht. Hierbei gilt die *Markov-Annahme*[4], dass der Nachfolgezstuand s_{t+1} nur vom aktuellen Zustand s_t und der gewählten Aktion a_t abhängt. Ziel eines Agenten ist es die Strategie p zu finden, die die Belohnungen maximiert. Die Strategie beschreibt folgende Funktion:

$$p : s_t \rightarrow a_t(s_t)$$

Sie besagt welche Aktion a in welchem Zustand s zu wählen ist.

Da bei Reinforcement Learning davon ausgegangen wird, dass der gesamte Zustandsraum bekannt ist und im Speicher abgelegt werden kann, sind große Zustandsräume problematisch. Um dieser Problematik auszuweichen, werden die zu umfangreichen Zustandsräume mit einem geeigneten Verfahren approximiert (z.B. unter Verwendung neuronaler Netze). Im Kontext des FAUST-Projektes wurde im Rahmen einer Ausarbeitung von Ivo Nikolov (Nikolov (2010)) bereits Reinforcement Learning mit Approximierung von Zuständen erfolgreich eingesetzt.

[4] *Markov property*: Nach dem russischen Mathematiker Andrey Markov

3 Bereich der modellbildenden Lernverfahren: *Dyna*

Beim modellfreien Reinforcement Learning wird oft auch von *lernenden* Methoden gesprochen. Wird hingegen mit Modellen gearbeitet, spricht man von *planenden* Methoden. Mit einem Modell der Umwelt umschreibt man bei Lernverfahren alles, was einem Agenten hilft, Vorhersagen über die Wirkung von Aktionen auf die Umwelt zu treffen.

3.1 Analyse und Erklärung von *Dyna*

Während sich Reinforcement Learning normalerweise mit direktem „Trial & Error"-Lernen beschäftigt, beschreibt Sutton (1991) bei *Dyna*, wie Reinforcement Learning mit einem gelernten Modell der Welt erweitert werden kann und wie mit dessen Hilfe Planung möglich ist. Planung meint hier das Lernen ohne direktes Ausprobieren. Vielmehr folgt es dem Ansatz „*trying things in your head*" (Sutton (1991)).

REPEAT FOREVER:

1. Observe the world's state and reactively choose an action based on it;
2. Observe resultant reward and new state;
3. Apply reinforcement learning to this experience;
4. Update action model based on this experience;
5. Repeat K times:
   - 5.1 Choose a hypothetical world state and action;
   - 5.2 Predict resultant reward and new state using action model;
   - 5.3 Apply reinforcement learning to this hypothetical experience.

Abbildung 3: *Dyna*-Algorithmus
Sutton (1991)

Der *Dyna*-Algorithmus ist in Abblidung 3 als Pseudo-Code dargestellt. Die Schritte eins bis drei entsprechen dem normalen Reinforcement Learning wie es in Kapitel 2 dargestellt ist. In Schritt vier wird die gemachte Erfahrung auf ein *action model* angewendet. Dieses *action model* ist eine *black box*, die als Eingabe ein Zustands-Aktions-Paar erhält und eine Vorhersage über den nächsten Zustand ausgibt. Das Innenleben dieser *black box* kann als Funktion beschrieben werden:

$$f(s_t, a_t) \rightarrow (s_{t+1}, r(s_t, a_t))$$

Diese Funktion ordnet jedem Zustands-Aktions-Paar ein Nachfolgepaar zu. Hierbei ist zu beachten, dass falls mit Wahrscheinlichkeiten gearbeitet wird, jede Aktion mehrere mögliche Folgezustände haben kann. Ein *action model* soll das Verhalten der Umwelt nachahmen. Schritt 5 führt nun Reinforcement Learning auf den Ergebnissen des *action models* aus, d.h. es wird k-mal ein hypothetisches Zustands-Aktions-Paar gewählt, dem *action model* übergeben und Reinforcement Learning auf dessen Ergebnisse angewendet. Dieser Schritt ist der eigentliche Planungsprozess. Darüber wie k zu wählen ist bzw. welche Zustands-Aktions-Paare zu wählen sind, gibt es viele verschiedene Ansätze, jedoch keine einheitliche Meinung. In ihrem Paper beschreiben Amato und Shani (2010), dass bereits besuchte Zustands-Aktions-Paare als Eingabe genutzt werden können und mit einer Lernrate von $k = 25$ gute Ergebnisse zu beobachten sind.

3.2 Diskussion der *Dyna*-Architektur

Auf *Dyna* bauen viele weitere Verfahren auf (u.A. Amato und Shani (2010)) und es hat sich herausgestellt, dass diese Architektur eine gute Grundlage für Erweiterungen von Reinforcement Learning ist. Mithilfe der *Dyna*-Architektur lassen sich mit wenigen realen Versuchen und einer gewissen Anzahl simulierter Versuche in jedem Schritt eine deutliche Beschleunigung des Lernalgorithmus bzw. ein deutlich schnelleres Konvergieren beobachten.

Da *Dyna* schon durch viele Personen in der Wissenschaft verwendet und darauf aufgebaut wurde, muss darauf geachtet werden, dass Rücksicht auf aktuelle Ergebnisse dieser Arbeiten genommen wird. Hierdurch kann vermieden werden, dass neue vielversprechende Entwicklungen nicht übersehen werden.

4 Lösungsansatz für den Einsatz von *Dyna* im FAUST-Kontext

Die Fahrzeuge des FAUST-Projektes bieten verschiedene Sensoren zum Erfassen ihrer Umwelt. Die Informationen, die diese Sensoren liefern, dienen als Basis für die Diskretisierung der Zustände des Agenten. Die folgenden Sensoren werden benötigt, um den Zustand bestimmen zu können:

- Inkrementalgeber an Vorderrädern zur Geschwindigkeitsbestimmung
- ggf. Beschleunigungssensoren

Der FAUSTcore (FAUST (2011)) besteht aus diversen Algorithmen, die unter anderem aus Kamera- und Ultraschallsensoren den Streckenverlauf sowie die Position des Fahrzeuges auf der Fahrbahn bestimmen können. Diese Informationen fließen zusätzlich in den Zustand des

Agenten ein, um die Informationsbasis für die Entscheidungsfinden so detailiert wie möglich zu gestalten.

Um Einfluss auf die Umwelt des Fahrzeugs nehmen zu können, stehen verschiedene Aktoren und Steuereinheiten zur Verfügung. Darunter Vorgabe des Lenkwinkels, der Geschwindigkeit oder der Bremskraft. Aus den verschiedenen Stellwerten dieser Aktoren ergeben sich die möglichen Aktionen, die der Agent ausführen kann.

Da es sich bei den verschiedenen Informationsquellen und Aktoren um ein hochdimensionales Problem handelt, müssen diese ggf. approximiert werden, um Reinforcement Learning auf das gegebene Problemfeld anwenden zu können. Mit einem geeigneten Verfahren zur Approximierung können dann aus den verschiedenen Informationen kontinuierliche Zustands- und Aktionsräume diskretisiert werden.

4.1 Exploration vs. Exploitation

In der Fachliteratur wird häufig von der Balance zwischen Exploration[5] und Exploitation[6] gesprochen. Die dort beschriebenen Probleme gilt es auch für den zu konzipierenden Lernalgorithmus im FAUST-Kontext zu berücksichtigen. Der Algorithmus, der zum Lernen der optimalen Geschwindigkeit zum Einsatz kommt, muss so flexibel sein, dass der Grad der Exploration angepasst werden kann. Exploration bedeutet, dass nicht immer solche Aktionen gewählt werden, die laut Strategie die günstigsten sind, sondern mit einer bestimmten Wahrscheinlichkeit auch unbekannte Aktionen versucht werden, um den Aktionsraum zu erkunden. Dies wird häufig über einen ϵ-Faktor in der Strategie-Funktion bestimmt. Der ϵ-Faktor ist definiert durch $0 < \epsilon < 1$ und beschreibt, mit welcher Wahrscheinlichkeit nicht die optimale, sondern eine zufällige Aktion gewählt wird zu der noch keine Belohnung beobachtet wurde. Ohne den Einsatz eines solchen Verfahrens kommt es häufig dazu, dass der Agent nach einigen Lernepisoden immer wieder die gleichen Aktionen wählt, da sie momentan als günstigste erscheinen. Da zu diesem Zeitpunkt jedoch meist nur ein Bruchteil der möglichen Aktionen versucht wurde, werden eventuell günstigere Aktionen niemals entdeckt.

Beim Carolo-Cup besteht die Möglichkeit, einige Runden ohne Wertung auf dem Parcours zu absolvieren. Diese Phase kann mit einem hohen ϵ-Faktor genutzt werden, um möglichst breit den Aktionsraum zu erkunden und bei der Wertung mit einem $\epsilon = 0$ nur noch solche Aktionen zu wählen, die sich während der Lernphase als günstigste herausgestellt haben. Wie genau ϵ zu wählen ist, kann meist nur durch Testen und Auswerten der Lerngeschwindigkeit herausgefunden werden.

[5] engl. für Erkundung
[6] engl. für Verwendung

4.2 Simulation des Lernalgorithmus mit TORCS

Die in 1.2 beschriebene Simulationsumgebung bietet eine adäquate Basis, um den zu konzipierenden Lernalgorithmus zu testen, ohne dabei eines der FAUST-Fahrzeuge zu gefährden. Gerade bei den ersten Tests könnte es passieren, dass Aktionen gewählt werden, die das Fahrzeug unkontrolliert von der Fahrbahn abkommen lassen. Im Labor von FAUST besteht zwischen Fahrbahnrand und Wänden keine oder nur eine geringe Auslaufzone, was solche Fehler nicht zulässt.

Die beiden FAUST-Fahrzeuge für den Indoor-Bereich werden von mehreren Projekten in Anspruch genommen und stehen nicht jederzeit für Versuche zur Verfügung. Hier bietet die Verwendung von TORCS ebenfalls den Vorteil, Tests von Veränderungen im Lernalgorithmus beliebig durchführen zu können.

4.3 Risiken und Probleme

Das Themengebiet des Reinforcement Learning besteht bereits seit langer Zeit und bietet somit eine Vielzahl von Fachaufsätzen, wissenschaftlichen Publikationen und Büchern. Diese Flut an Informationen gilt es zu sichten und auszuwerten, um bereits bestehende Ansätze nicht zu übersehen, die eventuell den Erfolg des Projektes steigern oder beschleunigen könnten.

Reinforcement Learning beruht wie alle maschinellen Lernverfahren auf mathematischen Formeln und Gleichungen, die zum Teil über Jahre der Forschung entstanden sind. Diese gilt es zu durchdringen und vollständig zu verstehen, um einen fehlerfreien Algorithmus für das FAUST-Projekt entwickeln und einsetzen zu können.

TORCS bietet zwar eine gute Basis, um Reinforcement Learning darauf entwickeln und testen zu können, aber die Hardware, die TORCS zur Verfügung steht übersteigt die Hardware der FAUST-Fahrzeuge bei weitem. Dies könnte dazu führen, dass die Ausführungszeiten der konzipierten Algorithmen zur Approximation und Berechnung des modellbasierten Reinforcement Learning auf einem realen Fahrzeug zu hoch wären. Dies hätte zur Folge, dass sie unbrauchbar wären.

In der Theorie bringt der modellbasierte Ansatz mithilfe von Reinforcement Learning viele Vorteile, jedoch ist nicht gesichert, ob sich hierdurch im realen Projekt wirklich eine Verbesserung des Fahrverhaltens herbeiführen lässt. Es gilt vorerst die Qualität der bestehenden Steuerung zu erreichen und im weiteren Verlauf des Projektes den Beweis zu erbringen, dass durch den Einsatz eines Lernalgorithmus tatsächlich eine höhere Geschwindigkeit erzielt werden kann.

5 Zusammenfassung und Ausblick

Das in dieser Ausarbeitung beschriebene Problem der optimalen Regelung der Geschwindigkeit eines Fahrzeugs im Kontext des FAUST-Projektes eignet sich gut für eine Realisierung mit Reinforcement Learning. Die vielen Vorteile die eine solche Implementierung gegenüber einer konventionellen Regelung hat, wurden mit Beispielen aufgezeigt. Auch die Nachteile bzw. Risiken und Probleme wurden beleuchtet.

In dieser Ausarbeitung wurde ein Lösungsansatz für die Verwendung der *Dyna*-Architektur zur Erweiterung von Reinforcement Learning beschrieben, der die Steuerung der FAUST-Fahrzeuge verbessern bzw. optimieren soll. In AW2 gilt es durch weiterführende Analysen der bestehenden Ansätze in der Literatur, diesen Ansatz zu detaillieren und zu vervollständigen. Während der Projektphase soll an der Implementierung eines geeigneten Algorithmus gearbeitet werden, der den genannten Ansatz verfolgt, um bereits erste Tests mithilfe der Simulationsumgebung TORCS durchzuführen.

Literatur

[Abbeel u. a. 2006] ABBEEL, Pieter ; QUIGLEY, Morgan ; NG, Andrew Y.: Using inaccurate models in reinforcement learning. In: *Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA : ACM, 2006 (ICML '06), S. 1–8. – URL http://doi.acm.org/10.1145/1143844.1143845. – ISBN 1-59593-383-2

[Alpaydin 2008] ALPAYDIN, Ethem: *Maschinelles Lernen*. Oldenbourg, 2008. – ISBN 3486581147

[Amato und Shani 2010] AMATO, Christopher ; SHANI, Guy: High-level reinforcement learning in strategy games. In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*. Richland, SC : International Foundation for Autonomous Agents and Multiagent Systems, 2010 (AAMAS '10), S. 75–82. – URL http://portal.acm.org/citation.cfm?id=1838206.1838217. – ISBN 978-0-9826571-1-9

[Carolo-Cup 2011] CAROLO-CUP: *Homepage Carolo-Cup*. 2011. – URL http://www.carolocup.de/

[FAUST 2011] FAUST: *FAUST Homepage*. 2011. – URL http://www.informatik.haw-hamburg.de/faust.html

[Hester und Stone 2009] HESTER, Todd ; STONE, Peter: Generalized model learning for reinforcement learning in factored domains. In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. Richland, SC : International Foundation for Autonomous Agents and Multiagent Systems, 2009 (AAMAS '09), S. 717–724. – URL http://portal.acm.org/citation.cfm?id=1558109.1558111. – ISBN 978-0-9817381-7-8

[Jong und Stone 2007] JONG, Nicholas K. ; STONE, Peter: Model-based function approximation in reinforcement learning. In: *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA : ACM, 2007 (AAMAS '07), S. 95:1–95:8. – URL http://doi.acm.org/10.1145/1329125.1329242. – ISBN 978-81-904262-7-5

[Jong und Stone 2008] JONG, Nicholas K. ; STONE, Peter: Hierarchical model-based reinforcement learning: R-max + MAXQ. In: *Proceedings of the 25th international conference on Machine learning*. New York, NY, USA : ACM, 2008 (ICML '08), S. 432–439. – URL http://doi.acm.org/10.1145/1390156.1390211. – ISBN 978-1-60558-205-4

[Jung 2003] JUNG, Tobias: *Reinforcement Learning eine Kurzeinführung*. März 2003

[Kaelbling u. a. 1996] KAELBLING, Leslie P. ; LITTMAN, Michael L. ; MOORE, Andrew W.: Reinforcement Learning: A Survey. In: *CoRR* cs.AI/9605103 (1996). – URL http://arxiv.org/abs/cs.AI/9605103

[Lin 1992] LIN, Long-Ji: Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. In: *Mach. Learn.* 8 (1992), May, S. 293–321. – URL http://portal.acm.org/citation.cfm?id=139611.139620. – ISSN 0885-6125

[Nikolov 2010] NIKOLOV, Ivo: NFQ zur Optimierung eines Lenkungsreglers. (2010)

[Sutton 1988] SUTTON, Richard S.: Learning to Predict by the Methods of Temporal Differences. In: *Mach. Learn.* 3 (1988), August, S. 9–44. – URL http://portal.acm.org/citation.cfm?id=637912.637937. – ISSN 0885-6125

[Sutton 1991] SUTTON, Richard S.: Dyna, an integrated architecture for learning, planning, and reacting. In: *SIGART Bull.* 2 (1991), July, S. 160–163. – URL http://doi.acm.org/10.1145/122344.122377. – ISSN 0163-5719

[Sutton und Barto 1998] SUTTON, Richard S. ; BARTO, Andrew G.: *Reinforcement Learning: An Introduction*. The Mit Press, 1998. – URL http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html

[TORCS 2011] TORCS: *Homepage von TORCS, The Open Racing Car Simulator*. 2011. – URL http://torcs.sourceforge.net/

Abbildungsverzeichnis