Proc, SOAVE2000 - SelbstOrganisation von Adaptivem VErhalten, Ilmenau, Fortschritt-Berichte VDI, Reihe 10: Informatik/Kommunikationstechnik, Vol. 643, pp. 184-193, VDI-Verlag 2000

Reinforcement Lernen in komplexen Umwelten: Basiskonzepte und vergleichende Untersuchungen

Dimitrij Surmeli
Technische Universität Ilmenau, Fachgebiet Neuroinformatik
dima.surmeli@informatik.tu-ilmenau.de

Zusammenfassung

Zur erfolgreichen Anwendung des Reinforcement Lernen (RL) auf autonomen Robotern müssen die Probleme einer kontinuierlichen, verrauschten, instationären und unvollständig beobachtbaren Umwelt behandelt werden. Anhand des bekannten Stabbalanceproblems vergleichen wir verschiedene Agenten in einer kontinuierlichen Umwelt, die mit gezielten Änderungen instationär und nicht vollständig beobachtbar wird. Damit kommt sie den Anforderungen realer sensomotorischer Problemstellungen nahe. Die Systematik der Benchmarkuntersuchung lehnt sich an die für Benchmarks im überwachten Lernen an. Als Untersuchungsobjekte werden eine Reihe neuronaler Funktionsapproximatoren und RL-Algorithmen für die Agenten kritisch gesichtet und eigene Untersuchungen vor dem Hintergrund autonom operierender Roboter in realen Umwelten angestellt.

1 Einführung

Die Voraussetzung der Markov-Eigenschaft für das Reinforcement Lernens (RL) stellt dessen praktischer Anwendung auf autonomen Robotern viele Schwierigkeiten entgegen. Einige können mit neuronalen Funktionsapproximatoren entschärft werden, die sich wiederum mit vorhandenen Lösungen kombinieren lassen.

Zur Herausstellung besonders erfolgversprechender Konfigurationen für RL-Agenten auf Robotern werden die Agenten unter den Gesichtspunkten kontinuierlich-wertiger Inputs, sich ändernder Aktuatoreffekte und unzureichender Inputinformationen ausgewählt und anhand derselben Aufgabe verglichen. Als ein solches Experimentalszenario wurde das bekannte Stabbalanceproblem gewählt, da es einige der Phänomene der Robotik in realen Umwelten reproduzierbar abbilden kann. Die in diesem Szenario schon untersuchten Aspekte [10], [1], [15], z.B. des Rauschens, werden um eine Reihe weiterer Probleme ergänzt.

2 Reale Probleme und mögliche neuronale Ansätze

Im folgenden werden einige Probleme skizziert (Abb. 1), die Roboter in realen Umwelten antreffen können, sowie eine, sich daraus ergebende Auswahl neuronaler Funktionsapproximatoren. Aufgrund der besonderen Sensitivität sensorisch geführter autonomer Roboter, wie auch des Szenarios, gegenüber der Zustandsrepräsentation, wird in diesen ersten Betrachtungen ein Schwerpunkt auf Clusterer gelegt.

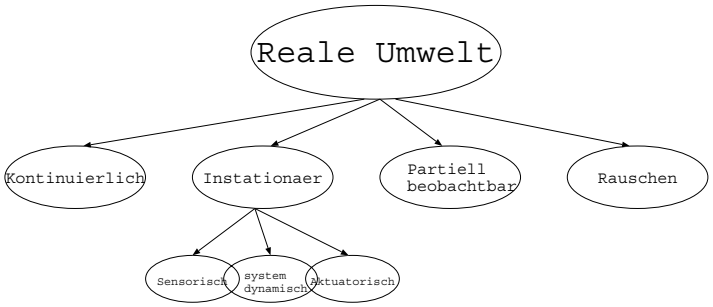

Abbildung 1: Schematische Darstellung möglicher Probleme in der realen Welt

2.1 Kontinuierliche Umwelt

Roboter müssen in einer kontinuierlichen Welt agieren, während die Mehrzahl der RL-Algorithmen von diskreten Zuständen ausgehen. Diese können durch eine vorgegebene (z.B. Tabelle, CMAC) bzw. adaptive Clusterung aus den kontinuierlichen Inputs erzeugt oder mit Hilfe von neuronalen Funktionsapproximatoren (FA) umgangen werden. Zur Auswahl für eine Untersuchung geeigneter FA ergibt sich eine Reihe von Kriterien: Eine erste Unterscheidung ist die in a) fixierende, neuronale Netze, die eine abgeschlossene Lernphase und damit eine stationäre Umwelt voraussetzen, aus der die Inputs für die Lernphase stammen, bzw. b) lebenslang lernfähige Netze. Letztere müssen das erste Dilemma zwischen Stabilität und Plastizität geeignet lösen.

Sowohl fixierende als auch lebenslang lernfähige Netze können die für viele RL-Algorithmen grundlegende Zustandswert- bzw. Zustands-Aktions-Wert-Funktion mittels eines direkten FA oder durch einen Zustandsclusterer mit nachgeschaltetem FA lernen. MLP's als direkte FA wurden zwar in anderen Studien [7] erfolgreich eingesetzt, waren in Voruntersuchungen aber den adaptiven Clusterern unterlegen. MLP's skalieren besser mit wachsender Anzahl von Eingangsgrößen, setzen aber eine Lernphase voraus, die ihren online-Einsatz behindert, und punktuelle Veränderungen in der Umwelt wirken sich global auf das schon erlernte Verhalten aus.

Adaptive Clusterer haben gegenüber vorgegebenen Clusterern den Vorteil, die Auflösung der Clusterung an die aktuellen Notwendigkeiten anpassen zu können. Der zusätzliche Aufwand zum Lernen der Clusterung lohnt in realen Umwelten nahezu immer.

Der Clusterer wiederum kann mittels des Clusterfehlers (datengetrieben) oder des Approximationsfehlers (handlungsgetrieben) adaptiert werden. Während der Adaptation muß ein zweites Dilemma, das der Unterscheidung zwischen Rauschen und wichtigen, also lernenswerten Ausnahmen gelöst werden. Statistische Clusterer eliminieren das Rauschen, indem über eine Vielzahl von Datenpunkten gemittelt wird. Dabei gehen wichtige Ausnahmen verloren, die beim Einschrittlernen erhalten werden. Letzteres erhält also die Ausnahmen, verbraucht jedoch unter Umständen unnütz viele Ressourcen.

2.2 Instationäre Umwelt

Unter diesem Begriff sollen verschiedene Probleme sich über die Zeit ändernder Bedingungen der Interaktion zwischen Agent und Umwelt zusammengefaßt werden. Hierbei ändert sich die Verteilungsdichtefunktion der sensorischen Inputs im Inputraum über die Zeit.

1. **Sensorische Instationarität**: Hier muß unterschieden werden zwischen einer echten Instationarität der Umwelt und der Änderung der Verteilungsdichte der erfahrenen Inputs durch das Erlernen einer guten Wertfunktion. Der Agent bewegt sich durch die abgeschlossene Exploration und adäquate Aktionen nur auf einer Teilmenge des ursprünglichen Inputsraumes. Dieses Problem betrifft den Clusteralgorithmus.

2. **Aktuatorische Instationarität**: Hierbei ändern sich Randbedingungen der Umweltbeeinflußung durch den Agenten, also die Wirkung der möglichen Aktionen. Die von uns betrachteten Agenten operieren nicht im tatsächlichen, kontinuierlichen Aktionsraum, sondern mit diskreten, bzw. quasidiskreten Aktionen. Dabei können sich die Zuordnungen zu bestimmten Aktionen unbeobachtet ändern (z.B. Radlagerschaden bewirkt zusätzliches Lenken oder ändert wirksame Kraft). Damit beziehen sich die bisher gelernten Wertfunktionen auf andere Aktionen, als wirklich ausgeführt werden und die Wertfunktion muß online adaptiert werden. Damit betrifft dieses Problem den RL-Algorithmus.

3. **Systemdynamische Instationarität**: Hierbei ändern sich die Zustandsübergangswahrscheinlichkeiten der Umwelt; unbeobachtete Änderungen von Systemparametern bzw. inneren Zustände der Umwelt verändern das Resultat einer ausgeführten Aktion. Dieses Problem betrifft sowohl den Cluster- als auch den RL-Algorithmus.

Wird ein datengetriebener Clusterer verwendet, der über den instationären Zeitraum noch plastisch ist, werden sich die Clusterzentren auf die aktuelle Teilmenge oder Trajektorie einschränken. Dies betrifft also insbesondere lebenslang lernfähige Clusterer. Das Problem kann effizient durch eine handlungsgetriebene Clusterung umgangen werden, die eine Verschiebung der Clusterzentren nur bei einer Verringerung des resultierenden Approximationsfehlers zuläßt ([1], [4]) oder aber durch geeignetes Versetzen von Neuronen ([6]). Alternativ läßt sich das Problem durch Vergrösserung des rezeptiven Feldes der Neurone lösen, wie bei FuzzyART ([3]).

2.3 Unvollständig beobachtbare Umwelt

Eine unvollständige Beschreibung des Zustandes der Umwelt im Inputvektor führt zu der Beobachtung, daß sich die Umwelt anscheinend nichtstationär verhält bzw. sogenannte versteckte Zustände aufweist. Obgleich dieses Problem Gegenstand intensiver Forschungen in den letzten Jahren war, sind dem Autoren keine RL-Algorithmen bekannt, die das Problem der *Partially Observable Markov Decision Processes (POMDP)* letztlich lösen würden. Ansätze bestehen in der Verwendung von 'Traces', wie in verschiedenen Temporal Difference Algorithmen als TD(λ) oder Q(λ) ([2],[8],[15]), der Zerlegung in mehrere Markov'sche Teilprozesse ([18]) oder dem Verschieben der Adaptation der Wertfunktion bis zum Eintreffen eines Rewards ([2]).

Schließlich lassen sich die beiden Probleme der instationären und der nicht vollständig beobachtbaren Umwelt als nicht modellierte Dynamik der Umwelt zusammenfassen. Allerdings ist eine entsprechende Modellierung aufgrund ihrer Komplexität illusorisch.

3 Struktur der Agenten

3.1 Verwendete Clusterer

In diesen Experimenten wird nur die aktuatorische Instationarität betrachtet und daher überwiegend Clusterer untersucht, deren Zentrenadaptation eingefroren wird.

Aufgrund biologisch motivierbarer topologischer Clusterung sowie guter experimenteller Erfahrungen ist das Neuronale Gas (NG, [9]) der Vergleichspunkt der Untersuchungen. Dies ist ein adaptiver optimaler Vektorquantisierer ohne feste Abbildungsdimensionalität und mit variabler Nachbarschaft beim Lernen der Zentren. Eine feste Anzahl von Zentren wird in nichtlinearer Abhängigkeit von ihrer Ähnlichkeit mit dem aktuellen Input zu diesem verschoben. Eine geeignete Steuerung der Nachbarschaft [4] realisiert auch eine handlungsgetriebene Clusterung.

Aus dem NG wurden mittels HEBB'*schen Wettbewerbslernens* das Growing Neural Gas (GNG, [5]) und die Dynamischen Zellstrukturen (DCS, [2]) entwickelt. Beide zählen aufgrund der

Effekte geeigneten Einfügens und Entfernens von Zentren zu den *perfekt topologieerhaltenden Karten* [1] und erlauben sowohl die Erweiterung um interpolierende Ausgabeschichten, handlungsgetriebene Clusterung, als auch eine Verfolgung nichtstationärer Inputs [6].

Ein Clusterer, dessen Zentren nicht eingefroren werden (lebenslange Lernfähigkeit) und der zum Einschrittlernen in der Lage ist, wird mit Fuzzy Adaptive Resonance Theory (FART, [3]) untersucht. Hier wird das Stabilitäts-Plastizitäts-Dilemma durch einen frei wählbaren Grad der Übereinstimmung zwischen aktuellem Input- und betrachtetem Zentrumsvektor gelöst. Stimmen z.B. bei einer wichtigen Ausnahme oder nichtstationären Inputs die Vektoren nicht überein, wird ein neues Zentrum angelegt, ansonsten wird das rezeptive Feld des 'resonierenden' Zentrums vergrößert. Verkleinerungen rezeptiver Felder sind dagegen nicht möglich.

Eine Kombination aus GNG und ART stellt das selbst entwickelte Growing Resonant Adaptive Neural Gas (GRANG, [13]) dar. Auf der Grundlage des GNG wird nach der Ermittlung eines den am besten Input repräsentierenden Zentrums zunächst dessen Übereinstimmungsgrad geprüft. Im Resonanzfall wird mit dem normalen GNG/DCS-Algorithmus weiterverfahren, ansonsten wird ein neues Neuron eingefügt. So wurden die wünschenswerten Eigenschaften dieser beiden Clustererkonzepte vereint.

3.2 Eingesetzte RL-Algorithmen

Das Q-Lernen (QL, [17]) wurde oft erfolgreich eingesetzt. Es wendet das Verfahren der *Temporal Differences* (TD, [14]) auf Zustands-Aktions-Paare an und kann so Adaptive Heuristic Critic und Adaptive Search Element (AHC und ASE, [15]) in einer Struktur vereinen. Zur Beschleunigung des Lernens durch unmittelbare Rückpropagierung erfahrener Wertänderungen auf vorher durchlaufene Zustände wird das QL um 'Traces' erweitert. Um eine Konvergenz zu garantieren, müssen diese Traces jedoch gelöscht werden, wenn eine explorative Aktion statt der besten (greedy) ausgeführt wird, damit die Werte nicht verfälscht werden.

Diese Einschränkung und den Maximum-Operator des QL sucht SARSA [12] aufzuheben. Bei diesem Verfahren ist allerdings die Konsistenz der Werte nicht gesichert [1].

Das Explorations-Exploitations-Dilemma soll nicht diskutiert, aber auf [16] verwiesen werden.

3.3 Untersuchte Agenten und Erwartungen

Hier sei die Liste der untersuchten Agenten zusammengefaßt: NG-Q, NG-Q(λ), NG-Q-SoftMax, GNG-Q, GNG-Q(λ), FART-Q, NG-SARSA, NG-SARSA(λ) und jeweils mit handlungsgetriebener Clusterung: GNG-Q, GNG-Q(λ), GRANG-Q(λ)

Eine einfache überwachte Schicht erlernt mit der Delta-Lernregel und dem Q-Fehler in einem Gewicht je einen Q-Wert für jede Aktion an jedem Zentrum. Die Lernrate ist immer konstant, somit mitteln die Werte über ein endliches Zeitfenster. Die Aktionsauswahl erfolgt mit der Boltzmann-Selektion. Alle Agenten erhalten einen einheitlichen Parametersatz bzgl. Lernraten, Anzahl der Zentren und Einfrieren der Adaptationen.

Sowohl QL als auch SARSA werden ohne Traces ($\lambda = 0$) und mit ($\lambda > 0$) hinsichtlich eines schnelleren Lernen der Werte untersucht, aber auch bzgl. deren Wirksamkeit bei POMDP: Agenten mit Traces sollten einen geringeren Leistungsabfall erleiden als Agenten ohne.

Darüberhinaus wird auch eine SoftMax-Variante des QL untersucht, bei der eine Adaptation aller Zustände abhängig vom Grad ihrer Ähnlichkeit mit dem aktuellen Zustand das QL beschleunigt. Allerdings könnte diese Modifikation zu Oszillationen der optimalen Aktion gerade in den Gebieten des Zustandsraumes führen, in denen eine Grenze zwischen zwei unterschiedlichen Aktionen verläuft, das rezeptive Feld aber Teile beidseits der Grenze einschließt.

Dieses Problem wird durch aktionsgetriebene Clusterung gelöst, bei der der Gradient des

Wertschätzfehlers die Adaptation der rezeptiven Felder bestimmt [1]. Daher werden Varianten von GNG und GRANG mit nützlicher Clusterung den datengetriebenen gegenübergestellt.

Die beim SoftMax-QL genutzte *räumliche Nähe* wird mit der *zeitlichen Nähe* der Traces verglichen. Wegen der Stetigkeit des Szenarios folgen ähnliche Zustände auch zeitlich aufeinander.

Wie erwähnt wirkt sich das Problem der aktuatorischen Instationarität besonders auf den RL-Algorithmus aus. Daher ist ein Vergleich von QL und SARSA informativ.

4 Umwelt für die Agenten

Da es in diesem Beitrag um einen prinzipiellen Vergleich der Leistungsfähigkeit verschiedener Algorithmen geht, haben wir zur Beschleunigung der Untersuchungen und zur Umgehung von hier irrelevanten Störungen eine Simulation einem realen Roboter in realer Umwelt vorgezogen. Als Aufgabe bzw. Umwelt haben wir das bekannte Problem der Stabbalance mit 2 möglichen Aktionen ($\pm$10N, Bang-Bang-Regelung) ausgewählt. Ein Wagen mit einem drehbar angebrachten Stab soll in einer Dimension so beschleunigt werden, daß der Stab balanciert wird. Für die Zwecke dieser Untersuchung erscheint es als besonders geeignet, da die Regelbarkeit des Systems kritisch von der Quantisierung des Zustandsraumes abhängt. Das System läßt sich mit zwei Differentialgleichungen (cf. [1]) und einer Eulerintegration realitätsnah simulieren und wird durch einen Vektor aus vier Variablen (Weg, Geschwindigkeit des Wagens, Auslenkungswinkel, Winkelgeschwindigkeit des Stabes): $(x, \dot{x}, \phi, \dot{\phi})$ beschrieben.

Jede Aktion des Agenten im zulässigen Bereich von $x \in [-2, 4m, ..., 2, 4m]$ und $\phi \in [-12°, ..., 12°]$ erzielt ein Reward von 0, ansonsten -1.

Anhand dieses Experimentalszenarios lassen sich die oben genannten Phänomene der Robotik gezielt, einzeln und in beliebiger Kombination, reproduzierbar und methodisch eindeutig simulieren. Die von der Umwelt an den Agenten übergebenen Zustandsvariablen sind kontinuierlich.

4.1 Simulation einer aktuatorischen Instationarität

Instationaritäten lassen sich hier leicht durch Veränderung der Systemparameter wie Reibungskoeffizienten, Längen, Massen oder der wirksamen Kräfte erzeugen. In unserer Untersuchung entschieden wir uns für eine extreme Variante der letztgenannten Möglichkeit, indem zu einem feststehenden Zeitpunkt die Richtung der Kraft umgekehrt wurde (so wird aus +10N -10N).

4.2 Simulation einer partiellen Beobachtbarkeit

Ähnlich simpel wird partielle Beobachtbarkeit realisiert: die berechnete Winkelgeschwindigkeit $\dot{\phi}$ wird durch gleichverteiltes Rauschen ersetzt. Würde eine Zustandsvariable ganz weggelassen oder konstant gehalten, wären die Lernverläufe für Clusterer und RL-Algorithmen, die sich exponentiell mit der Anzahl der Eingangsvariablen verhalten, nicht mehr vergleichbar.

5 Methodik des Benchmarks

Da uns für das RL keine Benchmarkmethodik analog der in der Mustererkennung gebräuchlichen ([11]) bekannt ist, schlagen wir eine an diese angelehnte Vorgehensweise vor, die gleichzeitig allgemeiner anwendbar ist als die bisher für das Stabbalanceproblem eingesetzte.

Über den gesamten zulässigen Zustandsbereich werden für eine statistisch sinnvolle Anzahl von Durchläufen je ein Trainings- und Validierungsdatensatz festgelegt, die gleichverteilt zufällig

Einsetzpunkte enthalten. Der Agent beginnt einen Versuch an einem Einsetzpunkt und verfolgt durch Interaktion mit der Umwelt eine eigenbestimmte Trajektorie, bis der Versuch an den Grenzen des Zustandsraumes und entsprechend positives oder negatives Reward endet. In einer Epoche wird dem Agenten der Trainingsdatensatz für eine feste Anzahl von Wiederholungen präsentiert, bevor mit dem Validierungsdatensatz der Fortschritt des Agenten kontrolliert wird.

Nach Erreichen eines Abbruchkriteriums wird der Agent fixiert (jegliches Lernen wird unterbunden) und ein Testdatensatz wird präsentiert, und die Leistungsfähigkeit des Agenten gemessen. Der Testdatensatz ist immer derselbe und enthält ebenfalls gleichverteilt zufällige Einsetzpunkte. Jeder Einsetzpunkt kommt in allen eingesetzten Datensätzen genau einmal vor.

In dieser Untersuchung stellen wir aber auch Ergebnisse der Agenten für die bislang bekannte Methodik vor. Dabei wird der Agent *immer* im Zustandsraum an $(0, 0, 0, 0)$ eingesetzt, verfolgt danach eine eigenbestimmte Trajektorie, bis er den zulässigen Bereich verläßt und das Reward erhält. Hierauf ist der Versuch beendet und der Agent wird neu eingesetzt. Ein Durchlauf erstreckt sich über eine bestimmte Anzahl von Gesamtschritten für alle Versuche, z.B. 500.000. Um mit dieser Methode statistische Aussagen treffen zu können, muß den Zustandsvariablen ein zufälliges Rauschen hinzugefügt werden, da die Simulation selbst ja deterministisch ist.

6 Ergebnisse

Im folgenden werden die Ergebnisse für die verschiedenen Agenten mit beiden Testmethoden vorgestellt. Dabei werden je Agent bis zu 50 Durchläufe ausgewertet. Infolge sehr hoher Simulationsdauer konnten nicht alle Agenten über 50 Durchläufe getestet werden. Für die Modifikationen an der Umwelt werden jedoch nur ausgewählte Agenten untersucht.

Die Bezeichnung der Agenten in Abbildungen und Tabellen folgt aus dem Clusterer, ggf. handlungsgetriebener Clusterung ('Nuetz'), dem RL-Algorithmus und ggf. dem verwendeten λ.

6.1 Neue Methode

In Tabelle 1 werden die Durchschnitte der Balancierschritte (erfolgreiche Schritte pro Versuch) über alle Durchläufe angegeben. Je höher der Wert, umso erfolgreicher der Agent. Als Robustheit wird die Standardabweichung der Anzahl der erfolgreichen Balancierschritte bezeichnet. Je kleiner der Wert, um so sicherer erzielt der Agent das angegebene Balancierergebnis.

Agent	∅ Balancierschritte	Robustheit	Prozent
NgQ0.0	178	81	45
NgQ0.8	136	56	41
NgSarsa0.0	108	87	80
NgSarsa0.8	115	64	56
NgQSoftMax	140	58	41
GngQ0.0	114	63	55
GngQ0.8	113	56	49
GngNuetzQ0.0	111	70	63
GngNuetzQ0.8	114	74	65
GrangNuetzQ0.8	109	64	59
FartQ	84	31	37
Zufall	13	1	8

Tabelle 1: Balancierergebnis der Agenten: der Agent mit NG und QL($\lambda = 0.0$) erzielt das beste Ergebnisse, während FartQ zwar ein schlechteres Ergebnis, dieses aber am zuverlässigsten erreicht.

Die Tabelle 1 und Abbildung 2 zeigen leichte Vorteile für den einfachsten Agenten, einen Neural Gas Clusterer mit QL ohne Traces. Es ist keine Beschleunigung durch Traces zu verzeichnen.

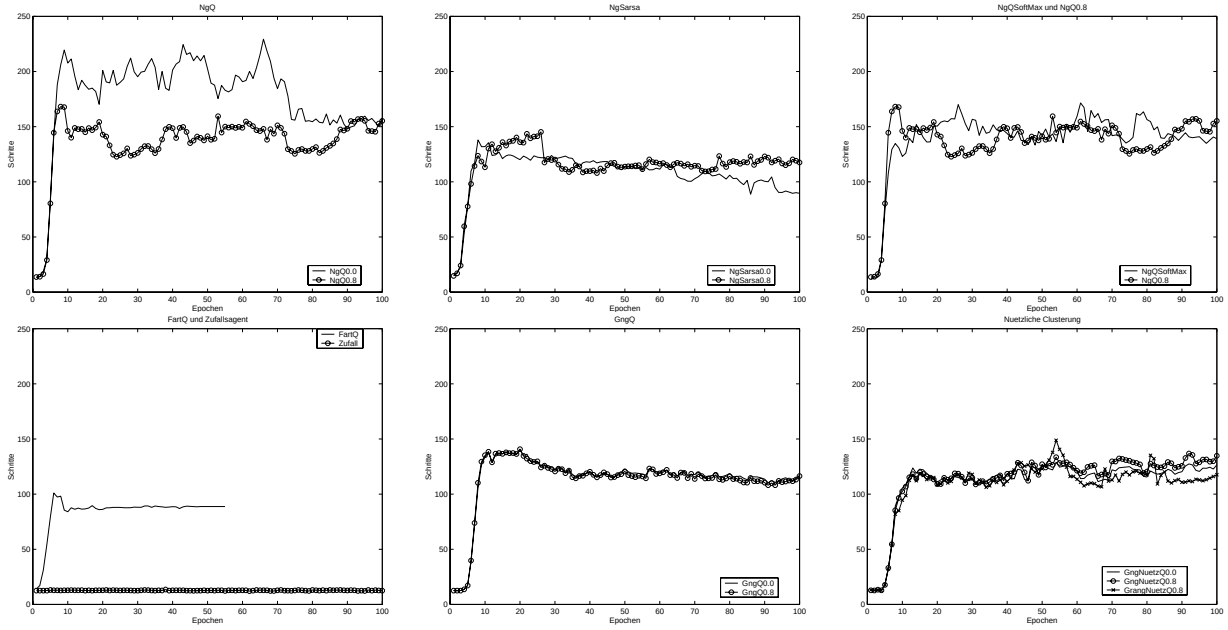

Abbildung 2: Gemittelter Verlauf der Anzahl erfolgreicher Balancierschritte

Dies widerlegt auch die Vermutung, SARSA würde aufgrund konsequenterer Nutzung der Traces schneller zu 'wahreren' Werten konvergieren. Offensichtich überwiegt die durch die Rückpropagierung 'falscher' Werte bei nicht-greedy Aktionen eingebrachte Inkonsistenz. Der Vergleich zwischen den RL-Algorithmen QL und SARSA geht zugunsten des QL aus. Anscheinend bestätigt sich auch die These, räumlich ähnliche Zustände folgten zeitlich aufeinander, da das QL mit Traces und das QL mit SoftMax sehr ähnliche Ergebnisse liefern.

Deutlich schlechtere Ergebnisse als die anderen Agenten liefert das FuzzyART mit einem QL ohne Traces. Dies führen wir auf die Stetigkeit des Systems zurück: Der Systemzustand ändert sich jeweils so gering, daß er auf dasselbe Neuron projiziert wird, welches daher sein rezeptives Feld so ausdehnt, daß die Clusterung zu grob für eine erfolgreichere Steuerung wird.

Unerwartet ist der langsame leichte Abfall der Balancierschritte nach einem Maximum für alle Agenten. Er ist aus der Kombination von Clusterung und zunehmender greedy-Policy zu erklären: wegen der greedy-Policy erhöht sich die Häufigkeit der in einem Cluster angetroffenen unkritischen Zustände und damit verbessert sich der Wert der schlechteren Aktion. Wird sie nun doch aus den vermeintlich selteneren Zuständen ausgeführt, führt das zum Mißerfolg.

6.2 Klassische Methode

Für einen direkten Vergleich sind in Abbildung 3 und Tabelle 2 die Ergebnisse der Agenten aus dieser Untersuchung denen aus der Literatur gegenübergestellt.

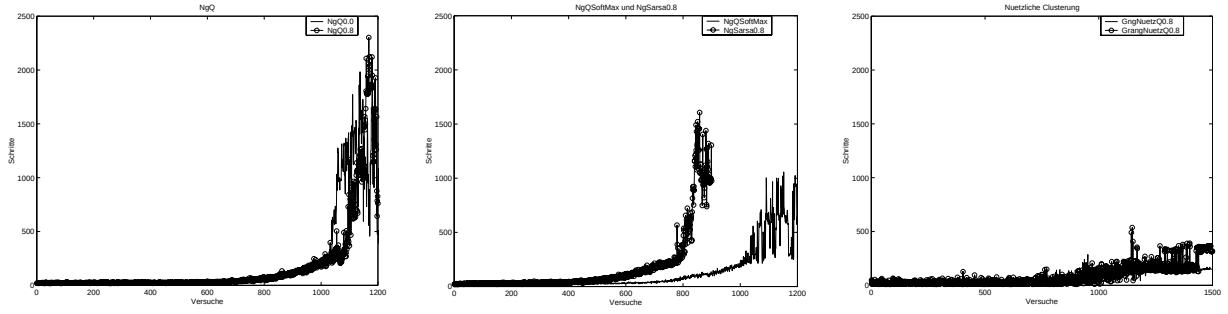

Abbildung 3: Balancierschritte nach der klassischen Methode für Stabbalance.

Quelle	Agent	Anz. Balancierschritte	∅ Versuche
[1]:	Modif. QL(0)	ca. 6.500	ca.950
	Modif. QL(1)	ca. 9.500	ca. 350
[10]:	BOXES	10.000	76,7
	AHC	10.000	56,3
	P-Trace	10.000	160,1
	Q-Trace	10.000	166,5
	QL	10.000	403,6
hier:	NgQ0.0	ca. 1.500	ca. 1.200
	NgQ0.8	ca. 1.500	ca. 1.200
	NgQSoftMax	ca. 1.000	ca. 1.200
	NgSarsa0.8	ca. 1.500	ca. 800
	GngNuetzQ0.8	152	1.200
	GrangNuetzQ0.8	294	1.200

Tabelle 2: Vergleich der Agenten nach der Klassischen Methode: es sollen, wenn möglich, 10.000 Balancierschritte pro Versuch in möglichst wenigen Versuchen erreicht werden.

Nicht zu übersehen ist der große Unterschied zwischen den hier erzielten und anderweitig berichteten Ergebnissen. Ein Grund ist der Ansatz, alle Agenten mit den gleichen Parametern ohne individuelle Feinabstimmung zu testen. Abstimmungen der Parameter ergaben in Nebenuntersuchungen Leistungsunterschiede in diesen Grössenordnungen, hätten aber die Vergleichbarkeit der Agenten eingeschränkt.

Der qualitative Vergleich der Ergebnisse zwischen beiden Methoden ergibt teilweise auch andere Aussagen; so ist zwar weiterhin keine Beschleunigung durch die Traces zu verzeichnen, jedoch ist SARSA schneller, wenn auch etwas schlechter. Weiterhin sind räumliche Ähnlichkeit (NgQ-SoftMax) und zeitliche Nähe (Traces, $\lambda > 0$) nach der klassischen Methode nicht äquivalent. Traces scheinen schneller zu lernen und bessere Ergbnisse zu liefern, wie eigentlich erwartet.

Ein wichtiger Grund für die Unterschiede zwischen den Ergebnissen für die beiden Testmethoden besteht darin, daß bei der klassischen Methode immer an $(0, 0, 0, 0)$ eingesetzt wird, der Clusterer damit gerade den Bereich um diesen Punkt sehr viel feiner auflösen kann und auch der RL-Algorithmus hier ausgiebig Möglichkeit hat, verschiedene Aktionsfolgen zu testen.

6.3 Aktuatorische Instationarität

Für diese Versuche wurden ca. in der 25. Epoche die Aktionen umgekehrt, was, wie erwartet, zu einem Abfall der Balancierschritte führte. Tabelle 3 und Abbildung 4 belegen die Ergebnisse.

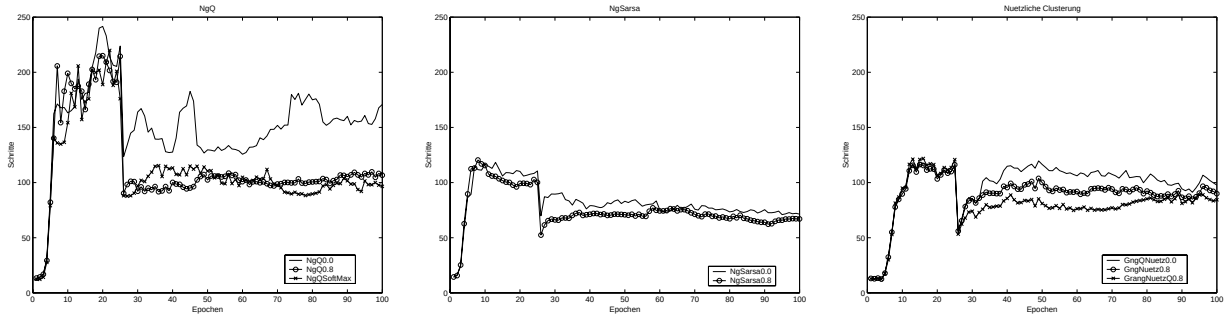

Abbildung 4: Gemittelte Balancierschritte der Agenten bei instationären Aktionen

Die Auswirkungen können mit der konstanten QL-Lernrate in den Trainigsversuchen schnell kompensiert werden, jedoch nicht bis zum ursprünglichen Niveau. Es läßt sich im Moment nicht genau unterscheiden, ob dies auf einen spezifisch 'instationären' Effekt oder auf den bereits in Abschnitt 6.1 konstatierten leichten Abfall nach einem Maximum zurückzuführen ist.

Der Vergleich zwischen den RL-Algorithmen geht wiederum zugunsten von QL aus.
Das schlechtere Ergebnis der nützlichen Clusterung war aufgrund der aus Vergleichsgründen na-

hezu eingefrorenen Clusterung zu erwarten. Hier ist das Ergebnis des Vergleichs der nützlichen Clusterer unerwartet: GNG-Q(0) kann ein besseres Retrainingsergebnis erzielen als GNG-Q(0.8) und GRANG-Q(0.8). Die Traces beschleunigten das Retraining wiederum nicht.

Agent	∅ Schritte	Robustheit
NgQ0.0	153	68
NgQ0.8	116	42
NgQSoftMax	113	54
NgSarsa0.0	78	37
NgSarsa0.8	114	69
GngNuetzQ0.0	98	66
GngNuetzQ0.8	92	39
GrangNuetzQ0.8	81	36

Tabelle 3: Balancierergebnis der Agenten bei instationären Aktionen: die Aktionen wurden in der 25. Epoche umgekehrt, die Ergebnisse über die Gesantzeit ermittelt.

6.4 Partielle Beobachtbarkeit

In Tabelle 4 und Abbildung 5 zeigt sich der erwartete rapide Abfall der Agenten mit einem völlig verrauschten $\dot{\phi}$. Die Agenten mit einem GNG-Clusterer sind besonders betroffen, da sie den hohen Wertfehler nicht durch Zentrenadaptation verringern können.

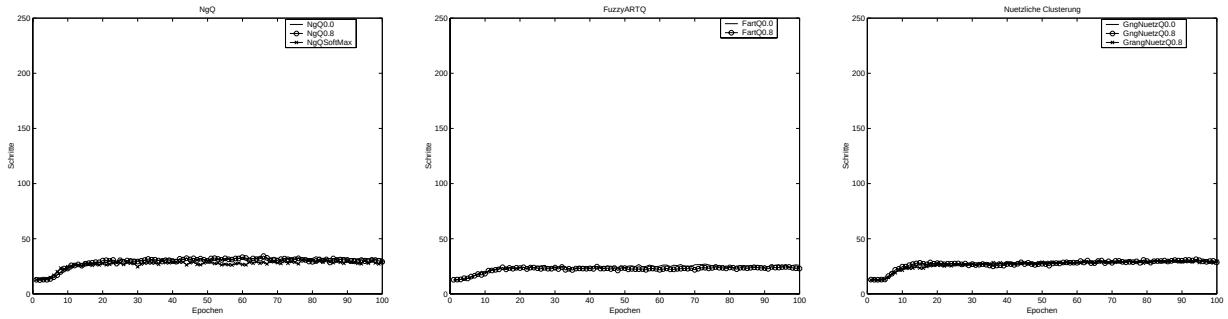

Abbildung 5: Balancierschritte der Agenten bei partieller Beobachtbarkeit

Agent	Schritte	Robustheit
NgQ0.0	29	7
NgQ0.8	29	7
GngNuetzQ0.0	27	6
GngNuetzQ0.8	27	6
GrangNuetzQ0.8	27	6
FartQ0.0	24	6
FartQ0.8	22	5

Tabelle 4: Balancierergebnis der Agenten bei partieller Beobachtbarkeit ($\dot{\phi}$ ist gleichverteiltes Rauschen). Alle Agenten fallen rapide, aber in der gleichen Rangfolge ab.

Auch tragen die Traces nicht zur Entschärfung der Probleme des POMDP durch schnelle Rückpropagierung der erfahrenen Werte und so korrekten Bias der Zustandsfolgen bei.

7 Diskussion

In diesem Beitrag berichteten wir über Vergleiche zwischen RL-Agenten anhand einer neuen Methodik, die teilweise anderen Veröffentlichungen widersprechende Ergebnisse erbrachte.

Wir beobachteten, daß RL-Algorithmen trotz vorhandener Lernerfolge sowohl in instationären als auch in partiell beobachtbaren Umgebungen insbesondere letztere große Probleme bereitet.

Weder Traces ($\lambda > 0$) noch handlungsgetriebene Clusterung konnten in dieser Untersuchung die in sie gesetzten Erwartungen erfüllen. SARSA kann keine Vorteile gegenüber QL nachweisen. Diese Ergebnisse fordern detailliertere Untersuchungen heraus.

Wir gingen von einheitlichen Parametern aus, die bestimmte Agenten bevorzugen könnten. Diese Ergebnisse müssen daher Ausgangspunkt genauerer Untersuchungen sein, besonders hinsichtlich Parameterabstimmungen für die Algorithmen. Allerdings ist ein Test aller denkbaren Agenten-Konfigurationen wenig sinnvoll, so daß wir künftige Untersuchungen wegen der Anforderungen an sensomotorische Systeme auf ART bzw. GRANG mit Modifikationen für instationäre Umwelten konzentrieren.

Literatur

[1] J. Bruske. *Dynamische Zellstrukturen: Theorie und Anwendung eines KNN-Modells*. PhD thesis, Technische Fakultät der Christian-Albrechts-Universität zu Kiel, 1998.

[2] J. Bruske, I. Ahrns, and G. Sommer. Practicing Q-Learning. In *Proc. of ESANN'96*, pages 25–30, 1996.

[3] G.A. Carpenter and S. Grossberg. *Pattern Recognition by Self-organizing Neural Networks*. MIT, 1991.

[4] R. Der and M. Herrmann. Efficient Q-Learning by division of labour. In *Proc. International Conference on Artificial Neural Networks - ICANN95*, volume II, pages 129–134, Paris, 1995.

[5] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7 (NIPS'95)*. MIT Press, 1995.

[6] B. Fritzke. Be busy and unique ... or be history - The utility criterion for removing units in self-organizing networks. In *KI-99: Advances in Artificial Intelligence*, pages 207–218. Springer, 1999.

[7] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

[8] John Loch and Satinder P. Singh. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *ICML'98,Int. Conf. on Machine Learning*, 1998.

[9] T. Martinetz and K. Schulten. A "neural gas" network learns topologies. In *Artificial Neural Networks*, pages 397–402. Elsevier Amsterdam, 1991.

[10] Mark D. Pendrith. On reinforcement learning of control actions in noisy and non-markovian domains. Technical Report UNSW-CSE-TR-9410, University of New South Wales, 1994.

[11] L. Prechelt. Proben1 - A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Universität Karlsruhe, 1994.

[12] G.A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University, UK, September 1994.

[13] D. Surmeli et al. Classification of 3d dendritic spines. In *Artificial Neural Nets and Genetic Algorithms(Proc.ICANNGA 97, Norwich, UK.)*, pages 129–132, 1997. Springer.

[14] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[15] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998.

[16] S. B. Thrun. The role of exploration in learning control. In *Handbook of intelligent control*, chapter 14. van Nostrand Reinhold, 1992.

[17] C. J.C.H. Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[18] M. Wiering and J. Schmidhuber. HQ-learning: Discovering markovian subgoals for non-markovian reinforcement learning. Technical report, IDSIA, 1996.