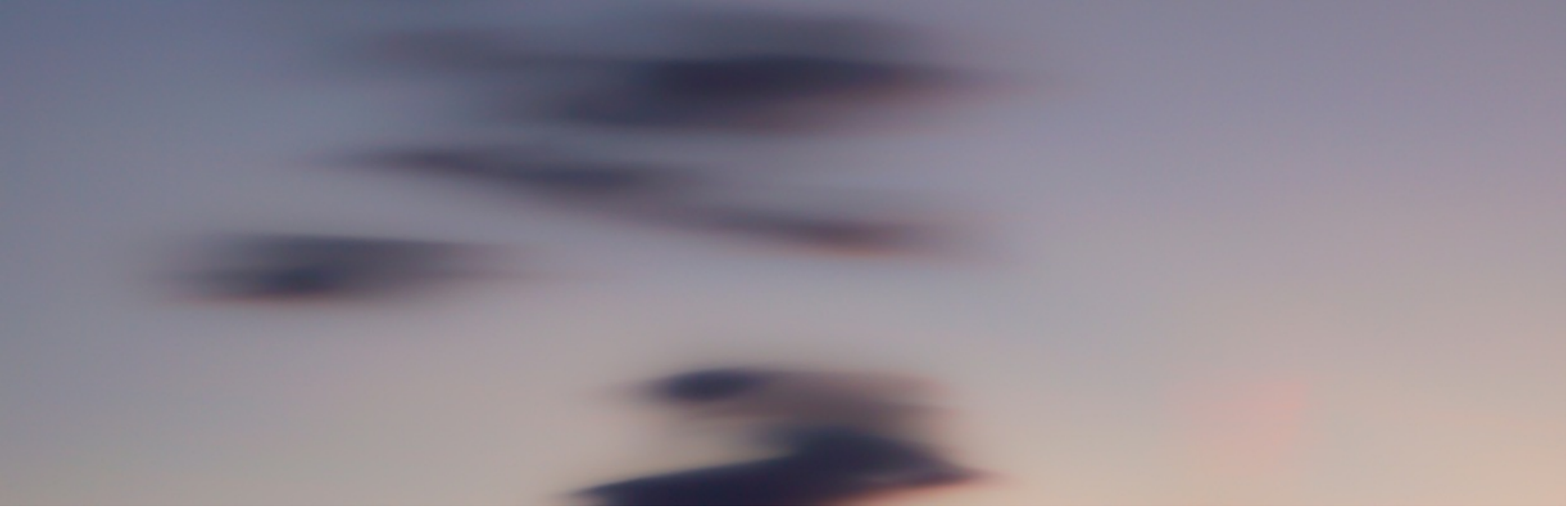


Tiefes Reinforcement Lernen auf Basis visueller Wahrnehmungen

Authors: Sascha Lange
Submitted: 10. December 2014
Published: 18. December 2014
Volume: 1
Issue: 1
Keywords: visuomotorische Lernprobleme, maschinelle Lernverfahren
DOI: 10.17160/josha.1.1.7

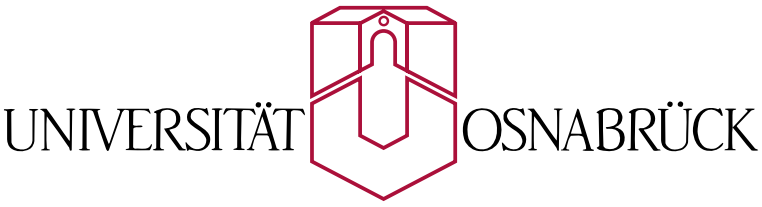
Tiefes Reinforcement Lernen auf Basis visueller Wahrnehmungen

Genehmigte Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

des Fachbereichs Mathematik / Informatik
der Universität Osnabrück

von Sascha Lange



Tag der mündlichen Prüfung: 7. April 2010
Erstgutachter: Prof. Dr. Martin Riedmiller, Universität Osnabrück
Zweitgutachter: Prof. Dr. Barbara Hammer, Universität Bielefeld

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

TABELLENVERZEICHNIS

1

Einleitung

In der Einleitung wird zunächst das in dieser Arbeit behandelte visuomotorische Lernproblem motiviert. Von der biologischen Lösung beim Menschen werden drei Teilprobleme abgeleitet, die auch von einem maschinellen Lernansatz zu lösen sind. Nach einer Diskussion erster bestehender maschineller Lernansätze zum visuomotorischen Lernen und der Identifikation von Problembereichen wird der eigene Ansatz umrissen und eingeordnet. Abgeschlossen wird die Einleitung mit einer Übersicht über die weitere Vorgehensweise und den Aufbau der Arbeit.

1.1 Handlungslernen auf Basis visueller Wahrnehmungen

Das Lernen auf Basis visueller Wahrnehmungen stellt existierende Verfahren des maschinellen Lernens immer noch vor große Herausforderungen. Dies gilt insbesondere im Bereich des Reinforcement Lernens [156] (auch "optimierendes Lernen", kurz RL), bei dem eine Sequenz von Handlungen nur durch Versuch und Irrtum in der Interaktion mit einem System erlernt wird. Zwar wurden mit Hilfe der wertfunktionsbasierten Reinforcement Lernverfahren in jüngster Zeit einige Erfolge an realen Systemen erzielt [47, 133, 134, 171], jedoch sind diese Methoden immer noch auf Problemstellungen mit sehr niedrigdimensionalen Zustandsräumen beschränkt. Aus diesem Grund ist eine direkte Anwendung des optimierenden Lernens auf hochdimensionale Bilddaten in der Praxis bisher ausgeschlossen. Dennoch wäre es natürlich in vielen Anwendungsgebieten des optimierenden Lernens im Bereich der Robotik, der autonomen Fahrzeuge und der Prozesssteuerung wünschenswert, auf Kameras als inzwischen preiswerte und universelle Sensoren zurückgreifen und direkt aus den Bildern eine Strategie zur Steuerung eines Systems erlernen zu können – ganz ohne die Anleitung und das Zutun eines Menschen. Genau hier setzt die vorliegende Arbeit an und leistet einen zentralen Beitrag mit der Entwicklung und Erprobung eines Verfahrens, das in der Lage ist, ohne vorgegebene Bildverarbeitung und Merkmalsextraktion Problemstellungen des Reinforcement Lernens direkt auf Bildern zu lösen. Zum Einsatz kommen tiefes Lernen [12, 61] und effiziente batch RL-Verfahren [39, 134].

1.2 Biologische Systeme

Für ein besseres Verständnis der Komplexität des Lernens in solchen visuomotorischen Aufgabenstellungen – auf der Basis visueller Wahrnehmungen muss eine Sequenz von Handlungen erlernt werden – lohnt sich ein Blick darauf, wie der Mensch selbst diese Probleme angeht. Allerdings dürfen hierzu nicht nur die Vorgänge beim unmittelbaren Lösen einer konkreten Aufgabe – wie zum Beispiel das Fangen eines Balles – betrachtet werden. Man muss schon wesentlich früher ansetzen, wenn verstanden werden soll, was alles zur erfolgreichen Lösung dazugehört. Denn tatsächlich geht der Mensch ein solches Lernproblem niemals ohne Vorwissen an. Er greift vielmehr auf ein bereits voll ausgebildetes visuomotorisches System zurück, das in einem viel länger dauernden Prozess entwickelt wurde und schon einen beträchtlichen Teil der Lösung mitbringt.

So besitzt der Mensch mit seinem Zentralnervensystem ein komplexes System von Vorverarbeitungs- und Abstraktionsebenen. Die Verarbeitung beginnt in den rezeptiven Feldern der Ganglien der Retina, zieht sich über den Corpus Geniculatum Laterale und die Orientierungsdominanzkolumnen im primären visuellen Kortex bis hin zu höheren kortikalen Regionen [62, 72], von wo dann über den motorischen Kortex und das Kleinhirn bis zu den Rückenmarksreflexen hinabgestiegen wird, um Bewegungen willentlich auszulösen und flüssig zu steuern. Diese Strukturen des Zentralnervensystems wurden in Äonen der phylogenetischen Entwicklung evolutionär angelegt [50]. Gleichzeitig gibt es aber auch Belege für eine Herausbildung und Verfeinerung von Verarbeitungseinheiten und -fähigkeiten in der postnatalen ontogenetischen Entwicklung [23, 32, 157] und damit Zeichen für "echtes Lernen" in der Interaktion mit der Umwelt. Dazu gehören die Herausbildung von rezeptiven Feldern auf der Retina [157] und hoch spezialisierter Orientierungsneuronen im visuellen Kortex [23] wie auch die Verfeinerung der Handlungssteuerung im Cerebellum [31]. In der frühkindlichen Entwicklungsphase spielt die Beobachtung und Interaktion mit der Umwelt dabei eine ganz entscheidende Rolle [31, 73, 89, 164]. Man denke zum Beispiel an einen Säugling, der seine Hände anfangs weitestgehend ziellos in seinem Gesichtsfeld bewegt und mit der Zeit zu fixieren und zu verfolgen lernt [7]. In unzähligen Stunden des zunächst zufälligen, später zielgerichteteren Agierens und Beobachtens wird langsam ein – dann vollständig automatisiertes – Verständnis für den Zusammenhang zwischen den eigenen "Handlungsbefehlen" und den resultierenden Wahrnehmungen entwickelt. Ein Vorgang wie das Greifen von Objekten wird vom ersten Erlernen im Säuglingsalter an in der andauernden Erprobung und Beobachtung weiter verfeinert, bis schließlich zum Ende der ersten Lebensdekade vollkommen sicher und zielgerichtet zugegriffen und der Ball gefangen werden kann [78, 148].

Die generelle Strukur der Verarbeitung im Zentralnervensystem und das Lernen selbst sind beim Menschen zwar angeboren und werden durch einen "externen" (evolutionären) Prozess optimiert, aber einige lebensnotwendige Fähigkeiten werden ganz offensichtlich erst nach der Geburt vollständig ausgebildet. Dabei werden nicht nur die eigentlichen Handlungssequenzen, sondern auch viel grundlegendere Aspekte visuomotorischen Handelns erlernt. So können für das visuomotorische Lernproblem insgesamt

drei Teilprobleme identifiziert werden, die prinzipiell auch von technischen Systemen zu lösen sind:

Wahrnehmung Das System muss zunächst erlernen, relevante Details der Umgebung und insbesondere Veränderungen wahrzunehmen. (Mensch: Ausprägung und Verfeinerung rezeptiver Felder und Ortsneuronen, Lenken der Aufmerksamkeit, Fixation).

Modell Es muss ein Verständnis über die Auswirkungen eigener Handlungen auf die Umgebung und für den Zusammenhang mit wahrgenommenen Veränderungen entwickelt werden. (Mensch: Erkennen und Steuern der eigenen Extremitäten, Ansteuern von Zielen innerhalb des Gesichtsfelds).

Strategie Es muss erlernt werden, welche konkrete Handlungssequenz zu dem gewünschten Ziel bzw. zur optimalen Lösung des Problems führt. (Mensch: zielgerichtetes Greifen, Feinsteuerung von Bewegungsabläufen, zum Mund Führen, etc.)

Diese drei Teilprobleme bauen ganz offensichtlich aufeinander auf; so kann nicht zielgerichtet gehandelt werden, wenn verschiedene Situationen nicht unterschieden werden können und die Auswirkungen einzelner Handlungen nicht bekannt sind.

Jedes dieser drei Teilprobleme für sich war bereits vielfacher Gegenstand von Forschungen und Anwendungen des maschinellen Lernens. Für jedes Teilproblem und jede Problemstellung gibt es das passende Verfahren. Die gleichzeitige Lösung aller drei Teilprobleme und damit des vollständigen visuomotorischen Lernproblems allein durch den Einsatz von Methoden des Maschinellen Lernens bleibt aber weiter eine ungelöste Herausforderung.

Optimierendes Lernen wird in dieser Arbeit für die Lösung des Strategielernens und auch des Modelllernens eingesetzt. Zunächst löst es das dritte Teilproblem, das Erlernen einer optimalen Sequenz von Handlungen. Mit Hilfe modellfreier Verfahren wie dem Q-Lernen [170] kann ein (implizites) Modell auch gleichzeitig mit der Strategie auf Basis derselben Beobachtungen erlernt werden. Wie schon eingangs erwähnt, sind bestehende optimierende Lernverfahren aber auf Zustandsräume mit relativ wenigen Dimensionen d (typischerweise $d \ll 100$) beschränkt und können daher nicht so einfach auf hochdimensionale Bilddaten (typischerweise $d \gg 100$) angewendet werden. Genau hier liegt nun der Schwerpunkt der vorliegenden Arbeit: auf der Integration der Lösung des Wahrnehmungsproblems in den optimierenden Lernansatz.

1.3 Klassische Herangehensweise

Die klassische Lösung für das visuomotorische Lernproblem ist die gesonderte Lösung des Wahrnehmungsteilproblems in dem in der Abbildung 1.1 dargestellten, zweistufigen Lösungsansatz mit vorgeschalteter klassischer Bildverarbeitung ("Vorverarbeitung") und eigentlicher Problemlösung – und zwar nicht nur im Zusammenhang mit Methoden des optimierenden Lernens sondern auch beim überwachten Lernen. Zunächst wird mit

Methoden des maschinellen Sehens [42] eine deutliche Dimensionsreduktion der sensorischen Daten erreicht, indem aus den ikonischen Bildern die relevanten Informationen bzw. wichtigen Bildmerkmale[1] extrahiert und in kompakter Weise kodiert werden. Die Auswahl "relevanter" Informationen und der Art der Kodierung obliegt dem Entwickler und wird in Abhängigkeit von den Erfordernissen der konkreten Aufgabe getroffen, genauso wie die Auswahl der Methoden zur Extraktion dieser Informationen. Auf Basis des so konstruierten Merkmalsraums mit stark reduzierter Dimensionalität können dann Methoden des optimierenden Lernens erfolgreich zum Einsatz kommen. Es wird nur noch die Strategie durch den Lernansatz modifiziert, während die Merkmalsextraktion selbst unangetastet bleibt.

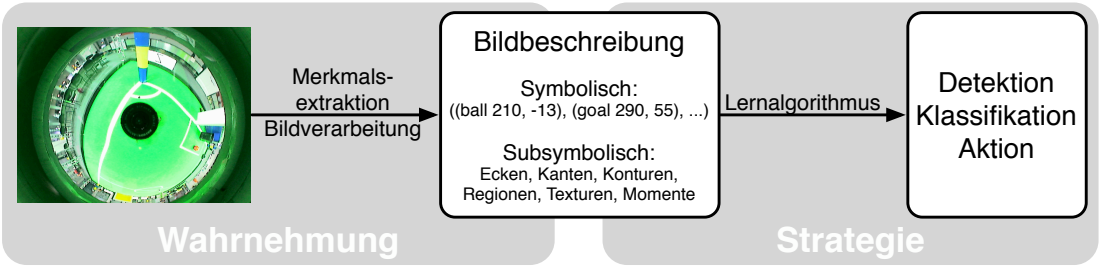


Abbildung 1.1: Klassische Vorgehensweise bei der technischen Lösung von visuomotorischen Lernproblemen. Aus den ikonischen Bilddaten wird zunächst eine kompaktere Repräsentation relevanter Informationen berechnet, auf deren Basis dann die Ausgabe des Systems durch überwachtes oder optimierendes Lernen bestimmt wird.

Im Rahmen dieses zweistufigen Ansatzes sind zwei unterschiedliche Varianten, eine explizit-symbolische und eine subsymbolische, zu identifizieren. Ein Beispiel für eine erfolgreiche Anwendung optimierenden Lernens auf ein visuomotorisches Lernproblem mit klassischer Bildanalyse und explizitem, symbolischen Weltmodell wird in [134] beschrieben. Der Ansatz von Groß et al. zum Erlernen eines Dockingverhaltens [56] ist dagegen ein typisches Beispiel für eine reale Anwendung optimierenden Lernens, in der zwar immer noch Vorverarbeitung betrieben, aber keine explizite, symbolische Kodierung berechnet wird. Als Zustandsraum des approximativen Reinforcement Lernverfahrens dient ein speziell entworfener, siebendimensionaler Vektorraum über $\mathbb{R}$, in dem Aktivierungen "globaler rezeptiver Felder" reellwertig kodiert werden. Im Vergleich zu dem vorhergehenden Ansatz mit explizitem, symbolischen Weltmodell ist diese Art der Vorverarbeitung zwar deutlich dünner, benötigt aber immer noch das Wissen eines Experten und ist speziell auf die Anwendung und die verwendete farbliche Kodierung der Umgebung zugeschnitten. [3, 5, 48, 49, 98, 108, 177] verfolgen ganz ähnliche Ansätze ohne explizites Weltmodell, aber mit jeweils ganz speziell zugeschnittenen Bildverarbeitungsmethoden und Merkmalsräumen.

[1] Mit "Merkmal" (engl. feature, image feature) ist meist eine globale (global feature) oder lokale (local feature), berechenbare Eigenschaft eines Bildes gemeint, wobei im Sprachgebrauch oftmals nicht sauber zwischen "markanter Stelle im Bild" und den Eigenschaften der Umgebung dieser Stelle unterschieden wird.

Soll gänzlich auf Hilfestellungen des Menschen bei der Verarbeitung der Bilder verzichtet werden, muss die Aufgabe als Ganzes, inklusive des Wahrnehmungsteilproblems, mit Methoden des selbstorganisierenden Lernens oder Reinforcement Lernens gelöst werden. Erst mit dem Verzicht auf jegliches Vorwissen in Form der Spezifikation der relevanten, zu extrahierenden Informationen oder der Angabe problemspezifischer Merkmalsextraktoren ist die Grundvoraussetzung für eine größere Autonomie und Flexibilität des Lösungsansatzes gegeben. Im folgenden Abschnitt werden erste Ansätze diskutiert, die in diesem Sinne auch das Wahrnehmungsteilproblem zum Gegenstand des Lernens machen.

1.4 Integrierte Herangehensweise

Einer der ersten, wenn auch überwachten Lernversuche, eine reale visuomotorische Lernaufgabe ohne klassische Vorverarbeitung, sondern direkt auf den Bildern zu lösen, war die an der CMU entwickelte neuronale Steuerung des autonomen Landfahrzeugs ALVINN [116, 117]. Korrekte Lenkradeinschläge wurden auf einem Satz von vorgegebenen Trainingsdaten überwacht erlernt. Dabei war eine auch für diese Arbeit wichtige Erkenntnis, dass es selbst in dieser vergleichsweise einfachen Klassifikationsaufgabe zu keiner guten Generalisierungsleistung des verwendeten dreischichtigen Netzes kam, wenn nicht die aufgenommenen Trainingsbilder (einige Tausend) über eine Analyse und synthetisch hergestellte Veränderung des Streckenverlaufs vervielfacht wurden [118]. Die evolutionäre Auswahl und Feineinstellung vorgegebener, klassischer Merkmalsextraktoren in evoVision [81] war ein weiterer Ansatz, eine aufgabenspezifische Lösung des Wahrnehmungsteilproblems zu erlernen. Für das überwachte Lernen entwickelt, wäre dieser Ansatz prinzipiell auch für eine Kombination mit Reinforcement Lernen geeignet, wenn die zufallsgesteuerte, evolutionäre Suche nach guten Merkmalsextraktoren nicht bei weitem zu ineffizient wäre.

Schon 1995 hatte Gordon die Idee, im Rahmen eines Anwendungsbeispiels seiner theoretischen Ergebnisse auch Verfahren des approximativen optimierenden Lernens direkt auf unvorverarbeitete Bilddaten anzuwenden [51], damals noch modellbasiert. Später wendete Ernst in einem ganz ähnlichen Ansatz auch modellfreie Reinforcement Lernverfahren auf ein Navigationsproblem direkt im Bildraum an [40]. Aufgrund des Einsatzes immer wiederkehrender und sehr künstlicher, synthetischer Bilder ohne Bezug zu natürlichen Bildern [40] sowie des völligen Verzichts auf Bildrauschen [40, 51], gibt es aber einige Bedenken hinsichtlich der Generalisierungsleistung und Anwendbarkeit dieser beiden Ansätze unter realen Bedingungen. Gegen eine gute Generalisierungsleistung spricht schon allein die große Anzahl von benötigten Trainingsbildern. So kommt bei Gordon von den weniger als 200 verschiedenen möglichen Observationen jede einzelne gleich mehrfach in den 5000 Trainingsbildern vor. Bei Ernst kommen auf jede der möglichen Observationen über 10 Trainingsbilder. So gilt es in diesen Versuchen lediglich, die gar nicht so zahlreichen, gleich mehrfach in den Trainingsdaten enthaltenenen Observationen eindeutig voneinander abzugrenzen und auf den dann eindeutig "benannten" Zuständen ein einfaches, diskretes Reinforcement Lernproblem zu lösen, für das prinzi-

piell keine Generalisierung über die Observationen nötig ist.

Dozono et al. verwenden einen interessanten, auf selbstorganisierenden Karten (engl. Self-Organizing Maps, kurz: SOM) [76] basierenden Ansatz, um eine Wertfunktion direkt auf wesentlich realistischeren, mittels OpenGL gerenderten Bildern zu approximieren [35]. Es gelingt ihnen, eine einfache, zu [56] sehr ähnliche Navigationsaufgabe zu erlernen. Allerdings wurden auch hier Bildrauschen und andere Störeinflüsse vernachlässigt und lediglich ein immediate-reward Problem [156] gelöst – der Agent erhält eine Belohnung, wenn er sich in Richtung des Ziels bewegt. Eine solche Gestaltung der Belohnungsstruktur (Stichwort: reward-shaping) ist wegen fehlender Informationen in der Realität aber nicht durchführbar. Außerdem wird von starken Einbrüchen der Ergebnisse berichtet [36], wenn Hindernisse hinzugefügt oder die künstlichen Farbkodierungen entfernt werden.

Im Gegensatz zu diesen drei Ansätzen, die die Wertfunktion direkt im Bildraum zu approximieren versuchen, verfolgen Jodonge und Piater [65, 66, 69] den vielversprechenderen Ansatz, die Wertfunktion in einem Merkmalsraum zu approximieren. Hierzu extrahieren sie hoch-diskriminative, auf dem Harris-Detektor [58] basierende visuelle Merkmale [55], ähnlich zu den wohl bekannteren und in [67] alternativ verwendeten SIFT-Merkmalen [93]. Anders als beim klassischen zweistufigen Ansatz wird bei ihnen der Merkmalsraum nicht gänzlich vorgegeben, sondern zumindest die Auswahl der für das Problem interessanten Merkmale einem in den Lernansatz integrierten Entscheidungsbaum überlassen. Ein Mangel ist allerdings, dass keine echte Generalisierung getestet wird. Zwar werden in den simulierten Labyrinthproblemen reale Bilder zur Repräsentation der Orte verwendet (Fotos), allerdings kommt bei jedem wiederholten Besuch eines der verwendeten diskreten Zustände (Labyrinthzellen) das exakt gleiche Foto zum Einsatz. Wie zuvor werden keinerlei echte Störungen simuliert, da die als einziges untersuchte Robustheit gegenüber leichten Rotationen [68] weniger eine Stärke des Lernverfahrens denn eine Eigenschaft des Merkmalsextraktors ist [55, 147].

Hinsichtlich der zunächst in der Simulation durchgeführten Labyrinthexperimente [65] ist weiterhin kritisch anzumerken, dass die gewählten Bilder von verschiedenen Objekten der COIL-100 Datenbank [107] jeweils bereits durch ein einziges der extrahierten Merkmale eineindeutige identifiziert und von allen anderen Bildern unterschieden werden können. Die Formulierung des Aufteilungskriteriums an einer Verzweigung des Entscheidungsbaums ist in dieser Situation aufgrund der fehlenden natürlichen Variation besonders einfach: Es kann auf direkte Identität der Deskriptoren getestet werden. In [65, 68] ist dann auch zu erkennen, dass die verwendeten Entscheidungsbäume nicht "ähnliche" (z.B. benachbarte) Situationen aus dem Zustandsraum schneiden und immer weiter aufgliedern, sondern in den meisten Ebenen des extrem unbalancierten Baumes – in [65] gar vollständig zu einer Liste einzelner Bilder degeneriert – genau ein einziges, eine bestimmte Situation eindeutig identifizierendes Bildmerkmal auswählen und so immer nur ein einzelnes Bild abtrennen. Dieses "Auswendiglernen" einzelner Bilder würde unter realen Bedingungen schon mit leichtesten Variationen in den Aufnahmen nicht mehr zum Erfolg führen und schließt eine Anwendung in der Praxis somit aus. In [67] wird zunächst der Versuch unternommen, diesen Ansatz durch den Einsatz von Binary

Decision Diagrams [21] an Stelle der Entscheidungsbäume in Richtung realer Probleme auszubauen. Zur ersten erfolgreichen Demonstration einer Generalisierung auf zumindest einige zuvor nicht gesehene Bilder führt dann aber erst der vollständige Verzicht auf jegliche Merkmalsselektion und direkte Approximation der Wertfunktion auf dem vollständigen Merkmalsraum [64].

Die beiden vorangehenden Abschnitte 1.3 und 1.4 zusammenfassend lässt sich feststellen, dass noch kein Verfahren existiert, das erfolgreich in realen Systemen oder zumindest unter realitätsnahen Bedingungen eingesetzt werden konnte und gänzlich ohne vorgeschaltete Bildverarbeitung und Vorwissen auskommt. Einem Einsatz an realen Systemen stehen vielmehr noch einige grundsätzliche Probleme hinsichtlich der Dateneffizienz (im Sinne der Anzahl für gute Ergebnisse notwendiger Beobachtungen) und Generalisierungsleistung entgegen.

So verzichten die Methoden, die direkt im Bildraum arbeiten, zwar komplett auf eine Vorverarbeitung der Bilder und benötigen in den Experimenten kein Vorwissen. Es ist aber fraglich, ob die im Zusammenhang mit den verwendeten Approximatoren berechnete Distanz zwischen Farbwerten einzelner Pixel überhaupt das geeignete Kriterium ist, um erfolgreich über Bilder ähnlichen Inhalts zu generalisieren und zustandsdefinierende Grenzen im Bildraum zu ziehen. Gordon und Ernst testen diese Verfahren lediglich in Problemstellungen, bei denen ein Auswendiglernen der Trainingsbilder prinzipiell möglich und keine Generalisierungsleistung auf ähnliche Bilder erforderlich ist. Dabei muss bis zum Erlernen einer erfolgreichen Strategie jedes einzelne der möglichen Bilder im Mittel viele Male beobachtet werden. In der klassischen Bildverarbeitung ist es zudem schon lange üblich, die Klassifikation von Bildern und Objekten auf Basis von Merkmalen vorzunehmen [71], da dies allgemein zu besserer Generalisierung führt.

Vielversprechender erscheint deshalb Jodonges Ansatz, die Wertfunktion auf einem von den Bildern konstruierten Merkmalsraum zu approximieren. Allerdings verlagert sein konkreter Ansatz mit der Merkmalsselektion nur einen Teil des Wahrnehmungsproblems in den Lernansatz und überlässt die Konstruktion und Extraktion der Merkmale weiter dem Entwickler. Zwar kann hinsichtlich der Experimente auf synthetischen Bildern auch der Vorwurf des Auswendiglernens erhoben werden und ein wirklicher Praxistest steht weiterhin aus, aber immerhin ließ sich dieser Ansatz bereits in Richtung realistischerer Simulationen erweitern. Anzumerken bleibt, dass auf Basis der beschriebenen Unterscheidung nach An- oder Abwesenheit von hoch deskriptiven, visuellen Merkmalen nur eine Teilklasse der visuomotorischen Probleme lösbar ist und zum Beispiel keine Probleme gelöst werden können, bei denen die Position der Merkmale im Bild für die Auswahl des Steuerungssignals entscheidend ist.

1.5 Zielsetzung und Validierung

Das in dieser Arbeit verfolgte Ziel ist es, das Wahrnehmungsteilproblem beim visuomotorischen Lernen vollständig in einen neuen Ansatz zum optimierenden Lernen auf Bildern aufzunehmen und dabei erstmalig auch eine Anwendbarkeit auf reale Probleme zu erreichen. Zum Kernproblem gehören hierbei die Dateneffizienz, die bei Ernst und

Gordon verloren gegangen ist, und die teilweise damit zusammenhängende Generalisierungsfähigkeit auf realen Bildern, die in keiner der vorgenannten Arbeiten in ausreichendem Maße belegt werden konnte. Damit der Ansatz an realen Systemen praktische Anwendung finden kann, sollte er zudem bestimmten Effizienz- und Stabilitätskriterien genügen. So können für die spätere Bewertung der Leistungsfähigkeit und Praxistauglichkeit ganz konkret drei wesentliche Kriterien benannt werden:

Stabilität Der Lernvorgang sollte eine gewisse Stabilität, wenn nicht Konvergenz, aufweisen, in dem Sinne, dass die Schätzung der optimalen Wertfunktion durch zusätzliche Erfahrungen und weitere Aktualisierungen tendenziell besser wird und nicht dramatisch schwankt.

Effizienz Der Ansatz sollte effizient sein, wobei hier unter der Effizienz des Lernens zuerst die Dateneffizienz verstanden wird. Wie viel Beobachtungen benötigt der Algorithmus, um eine zielführende oder optimale Strategie zu erlernen? Zu Vergleichszwecken kann die erreichte Strategiegüte nach n Interaktionen oder Episoden herangezogen werden. Dies ist ein wichtiges Kriterium an realen Systemen, da hier Beobachtungen nicht wie in der Simulation "gratis" zu haben sind, sondern echte Kosten verursachen.

Generalisierung In realen, kontinuierlichen Systemen können nicht alle Zustände besucht und erprobt weden. Vielmehr ist es notwendig, von den in der Trainingsphase beobachteten Übergängen auf ähnliche Situationen zu schließen. Gerade bei der Verwendung von realen Bildern ist diese Fähigkeit besonders wichtig, denn aufgrund des komplexen und störungsbehafteten Bildformationsprozesses ist jede Beobachtung auch identischer Zustände anders. Dies spielt auch für die Effizienz eine Rolle. Ein Algorithmus, der nur getätigte Beobachtungen "auswendig" lernt, ist weder praxistauglich noch effizient.

Vergleichen lassen muss sich das entwickelte Verfahren dabei in all diesen Aspekten immer mit dem robusten, praxistauglichen und effizienten zweistufigen Ansatz mit vorgegebener klassischer Bildverarbeitung – auch wenn das Mitlernen der Merkmalsextraktion natürlich zwangsläufig zu einem gewissen zusätzlichem Aufwand führen muss.

1.6 Zweistufiger, integrierter Ansatz: "Deep Fitted Q"

Nachdem die Grenzen bestehender Verfahren identifiziert sind und die Zielsetzung formuliert ist, stellt sich nun die Frage, wie die konkreten Ziele erreicht werden können. Die in dieser Arbeit verfolgte Idee ist es, auf den klassischen zweistufigen Ansatz aufzubauen, die Konstruktion des Merkmalsraums dabei aber vollständig in das Lernverfahren zu integrieren (siehe Abb. 1.2). Dies geschieht, indem zur Lösung des Wahrnehmungsteilproblems in der ersten Stufe mit den jüngst vorgeschlagenen tiefen Autoencodernetzen [60, 61] ein unüberwachtes Lernverfahren zum Einsatz kommt, das lediglich auf den in der Interaktion gesammelten Observationen arbeitet und gänzlich ohne weitere

Hinweise, wie zum Beispiel die Vorgabe von Trainingsbeispielen, auskommt. Tiefe Autoencodernetze sind für diese Aufgabe besonders geeignet, da sie in der Lage sind, aus Bilddaten automatisch die relevanten Informationen zu extrahieren und sehr kompakte, informationserhaltende Repräsentationen zu erzeugen.

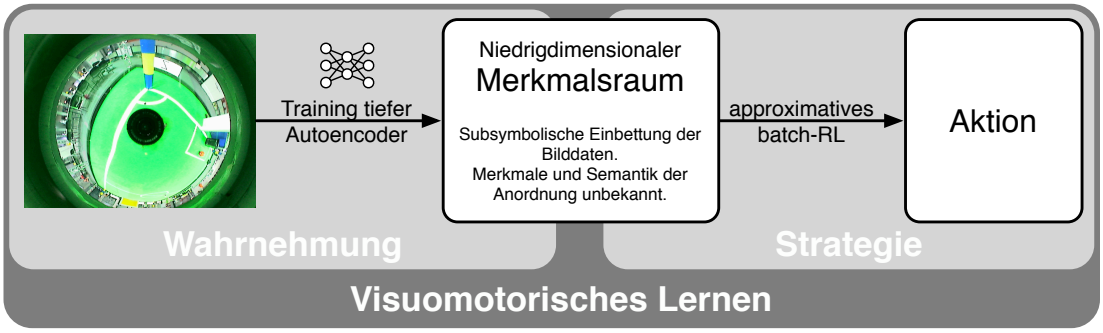


Abbildung 1.2: Eigener zweistufiger Ansatz, der ohne klassische Bildverarbeitung und ohne domänenspezifisches Vorwissen auskommt.

Die automatische Erzeugung von kompakten Merkmalsräumen mittels der Autoencoder wird in das speicherbasierte batch RL-Verfahren Fitted Q-Iteration [39] eingebettet, mit dessen Hilfe an Hand der im Merkmalsraum beobachteten Übergänge in der zweiten Stufe eine Wertfunktion approximiert werden kann. Batch RL-Verfahren sind dadurch gekennzeichnet, dass sie im Gegensatz zu reinen online Verfahren die aktuelle Beobachtung nicht nur für eine einmalige Verbesserung der Schätzung der Wertfunktion heranziehen, sondern zwischenspeichern und sie immer wieder zur inkrementellen Verbesserung der Schätzung verwenden. Aufgrund dieser in beiden Fällen vollzogenen Speicherung und Wiederverwendung der Daten passen batch RL-Verfahren und tiefe Autoencoder besonders gut zusammen und ermöglichen eine optimale Ausnutzung der am System gewonnenen Erfahrungen.

Überhaupt wird im Rahmen dieser Arbeit besonderes Augenmerk auf die effiziente Einbettung des Erlernens geeigneter Merkmalsräume in das optimierende Lernen und die Reduktion des dadurch potenziell entstehenden Mehraufwandes gelegt. Um die drei Stabilitäts-, Effizienz- und Generalisierungskriterien erfüllen zu können und an die Praxistauglichkeit des klassischen Ansatzes heranzureichen, sind Abstimmungen und Weiterentwicklungen auf allen Ebenen nötig, sowohl im Gesamtablauf – also beim Zusammenspiel der beiden Stufen – als auch bei den innerhalb der einzelnen Stufen zum Einsatz kommenden Lernverfahren:

- Einerseits hängt die Dateneffizienz des Algorithmus ganz entscheidend ab vom Zusammenspiel der beiden Stufen und dem genauen Ablauf beim Wechsel zwischen Verbessern der Einbettung und Approximieren der Wertfunktion. Erst durch die neuen Methoden zum tiefen Lernen und die Verfügbarkeit von batch Verfahren wird ein zum Erfolg führender, effizienter Ablauf ermöglicht, bei dem Optmieren der Strategie, Sammeln neuer Beobachtungen und Verbessern der Einbettung Hand in Hand gehen. Kapitel 3 widmet sich eingehend diesem Gesamtablauf und

liefert als einen wichtigen Beitrag Rechenzeit und Interaktionen einsparende Methoden zum Transfer des bisher gewonnenen Wissens zu einem neuen, verbesserten Merkmalsraum. Diese Methoden werden erst durch die Verwendung speicherbasierter RL-Verfahren ermöglicht.

- Andererseits sind der Lernerfolg und insbesondere die Stabilität und Generalisierungsleistung des entwickelten Algorithmus auch ganz entscheidend von Größe und Qualität der Merkmalsräume und von der Auswahl und Leistungsfähigkeit des im batch RL eingesetzten Funktionsapproximators und damit von der konkreten Realisierung der beiden Module des zweistufigen Ansatzes abhängig. Für eine optimale Lösung gilt es, die in den beiden Stufen zum Einsatz kommenden Verfahren auf die Anwendung in visuomotorischen Lernproblemen abzustimmen und einige grundsätzliche Probleme zu identifizieren und zu lösen. Wichtige Anhaltspunkte liefert dazu die theoretische Untersuchung des DFQ-Algorithmus in Kapitel 4, in dem auf Basis der Ergebnisse von Gordon [51, 53] und Ormoneit et al. [111–113] einerseits die Konvergenz des Algorithmus gegen einen eindeutigen Fixpunkt – unter bestimmten Bedingungen – nachgewiesen werden kann und andererseits wichtige Anforderungen an Merkmalsräume und Approximatoren formuliert werden.

Diese Fragestellungen in Bezug auf die Realisierung der beiden Stufen im hier entwickelten "Deep Fitted Q"-Algorithmus[1] (DFQ) werden in den beiden folgenden Abschnitten eingehender diskutiert.

1.7 Stufe 1: Wahrnehmung

Das tiefe Lernen wurde bisher hinsichtlich Bildverarbeitungsproblemen hauptsächlich in ganz klassischen Problemstellungen der Objektklassifikation [61] und Gesichtserkennung [24] untersucht. Insbesondere auf dem MNIST-Datensatz [88] wurde eine hervorragende Leistung nachgewiesen und es zeigte sich anderen Lernverfahren wie flachen neuronalen Netzen und Support Vector Machines deutlich überlegen [61, 88, 144]. Die in diesen Experimenten mittels tiefer Autoencoder automatisch erzeugten nichtlinearen Einbettungen in niedrigdimensionale Merkmalsräume wiesen zudem eine deutlich höhere Qualität auf als die mittels Hauptkomponentenanalyse (engl. Principal Component Analysis, PCA) [70] und Locally Linear Embedding (LLE) [138] aus den gleichen Daten gewonnene Einbettungen. Zudem ermöglichten sie höhere Klassifikationsraten eines nachgeschalteten Klassifikators [61]. Diese eindrucksvollen Ergebnisse ermutigen zu dem Versuch, auf der Basis tiefen Lernens einen Algorithmus zur Lösung des visuomotorischen Lernproblems zu entwickeln. Allerdings ist es aus drei Gründen trotz der publizierten Ergebnisse noch unklar, ob sich dieses Verfahren überhaupt für den hier geplanten Einsatz im visuomotorischen Lernen eignet:

[1] Die Namensgebung spielt auf die dem DFQ-Algorithmus zugrunde liegenden tiefen Autoencodernetze und Ernsts Fitted Q-Iteration an.

1. Eine offene Frage ist, ob in der Anwendung auf RL-Probleme eine vergleichbar gute Generalisierung erzielt werden kann, bei denen es zumeist um die Bestimmung der Position und Lage von einzelnen Objekten im Bild bzw. deren Verfolgung über eine Bildsequenz geht. Zum einen unterscheiden sich die Bilddaten in diesen Lokalisations- und Verfolgungsproblemen deutlich von denen in der Objektklassifikation, denn von einem Bild zum nächsten kommt es bei sich langsam bewegenden oder kleinen Objekten nur zu sehr geringen Änderungen. Diese kleinen Änderungen müssen dennoch zuverlässig erfasst, von unvermeidbarem, überlagernden Bildrauschen getrennt werden und Eingang in die erzeugte Repräsentation finden. Zum anderen wird für die Anwendung als Basis für die Wertfunktionsapproximation eine Kodierung der Bildinformationen in sehr viel weniger Dimensionen als in den Objektklassifikationsaufgaben angestrebt; typischerweise in zwei Dimensionen in Navigationsaufgaben gegenüber 30 bis 2000 auf MNIST. Einige zur Visualisierung erzeugte 2D-Einbettungen in [61] deuten zwar in diese Richtung, aber ob das Hindurchführen der relevanten Informationen durch diesen extrem schmalen Flaschenhals auch im RL-Zusammenhang in ausreichender Qualität gelingt, ist zunächst unklar.

2. Autoencoder werden auf die Rekonstruktion der angelegten Eingabe in der Ausgabeschicht trainiert. Gelingt das Erlernen guter Rekonstruktionen, bedeutet das aber noch nicht zwangsläufig, dass die Bilder dabei so im Merkmalsraum angeordnet werden, dass unterschiedliche Zustände im Sinne des Reinforcement Lernproblems an Hand der Merkmalsvektoren zweifelsfrei voneinander unterschieden werden können. Darüber hinaus stellt sich die Frage, ob RL-Verfahren aus der automatisch erzeugten Anordnung im Merkmalsraum einen Nutzen ziehen und über ähnliche Observationen verallgemeinern können. Eine solche Abstraktion ist zwingend erforderlich, da im Prinzip identische Situationen in der Praxis viele verschiedene Bilder hervorrufen können, die aber nur zu einem kleinen Teil im Training beobachtet werden können.

3. Ein weiteres Problem ist die Menge und Verteilung der Observationen im Observationsraum. In realen RL-Problemstellungen kommt es aufgrund der selbständig durchzuführenden Exploration des Zustandsraums nicht zu einer so gleichmäßigen "Abdeckung" des Observationsraums wie in den Klassifikationsproblemen (MNIST: 6000 Bilder je Klasse). Anfänglich liegen Observationen typischerweise um den Startzustand, später kommt es oft zu einer natürlichen Häufung um das Ziel und entlang der optimalen Trajektorie. Tatsächlich läuft eine gleichmäßige Abdeckung des Zustandsraums meist dem eigentlichen Lernziel zuwider. Gleichzeitig ist auch die Menge der für ein erfolgreiches Training erforderlichen Bilder von Bedeutung. Zwar wurden in den Klassifikationsexperimenten gemessen an der Netzgröße vergleichsweise wenige Daten verwendet, jedoch sind 60 000 Trainingsbilder für 10 Klassen (MNIST) in RL-Problemen immer noch viel zu viel. Um eine Einsetzbarkeit an realen Systemen zu ermöglichen, sollte das Training der Autoencoder auch mit einigen hundert bis wenigen tausend Bildern zum Erfolg

führen. Wichtig in Hinblick auf die anfängliche Exploration des Systems und das "Bootstrapping" des Kontrollers zu Trainingsbeginn ist auch die Frage, ob mit sehr wenigen Bildern und unvollständiger Abdeckung des Observationsraums bereits etwas gelernt werden kann, um die besuchten Zustände für eine erfolgreiche Steuerung der Exploration auseinander halten zu können.

Die Fragen eins und zwei betreffen im Wesentlichen die Generalisierungsfähigkeit aber auch die Stabilität des Lernvorganges auf den automatisch erzeugten Merkmalsräumen. Frage drei ist vornehmlich in Hinblick auf die zu erreichende Dateneffizienz im RL-Ansatz von Bedeutung. Müssen für die Strategie uninteressante Regionen exploriert werden, um eine gute, stabile Einbettung zu erzielen? Müssen wesentlich mehr Observationen gesammelt werden, als für das eigentliche Erlernen einer Strategie nötig sind? Und wie können die Anforderungen an Anzahl und Verteilung der Observationen deutlich reduziert werden?

Tatsächlich ist es so, dass eine direkte Anwendung Hintons tiefer Autoenconder mit schichtenweise voller Vernetzung zunächst in allen drei Belangen nicht zum gewünschten Erfolg führt. In der Bearbeitung dieser Fragen in Kapitel 5 wird dann mit eigenen ausführlichen Untersuchungen und Auswertungen und durch die Entwicklung einer für die Anwendung in den RL-Problemen geeigneten Netzarchitektur ein eigener, wichtiger Beitrag zu den bisher publizierten Ergebnissen geleistet. Zur Klärung der Fragen eins und zwei werden die erzeugten Merkmalsräume verschiedener Netze analyisert. Hinsichtlich der Frage drei werden die entwickelten Netzarchitekturen in Rekonstruktions- und Klassifikationsaufgaben auf verschieden großen, RL-typischen Datensätzen verglichen.

1.8 Stufe 2: Strategielernen

Bei den approximativen batch RL-Verfahren kommt eine große Bedeutung dem eingesetzten Funktionsapproximator zu, der mittels überwachten Lernens auf eine iterativ verbesserte Schätzung der optimalen Wertfunktion trainiert wird. Einerseits muss dieser Funktionsapproximator eine gute Generalisierungsleistung im kontinuierlichen Zustandsraum ermöglichen, andererseits muss er bestimmten Anforderungen genügen, um einen stabilen Lernvorgang zu ermöglichen. Viele beliebte Approximatoren wie neuronale Netze, Multigrids (z.B. CMAC) und selbst lineare Regression erfüllen diese Anforderungen nicht.

Da in der Kombination mit der automatisch erzeugten, und damit nicht voll zu kontrollierenden Einbettung weitere Quellen möglicher Instabilität vermieden werden sollen, wird der Einsatz eines – unter den in Kapitel 4 erarbeiteten Bedingungen – sicheren Verfahrens angestrebt. Ein solches Verfahren sind Approximatoren auf Basis von regulären Gittern, die den Zustandsraum in reguläre Hyperzellen mit achsenparallelen Zellwänden einteilen und allen in dieselbe Hyperzelle fallenden Zuständen den gleichen Wert zuweisen. Generalisierung findet hier nur lokal, innerhalb der Zellen statt und ist damit stark abhängig von der richtigen Lage der Zellgrenzen, die vorab von außen vorgegeben werden müssen und sich nicht an die zu approximierende Funktion anpassen. Eine optimale Einstellung dieser und anderer Parameter ist aber für die vorab unbekannten

und im Lernverlauf sich ständig ändernden Merkmalsräume nur schwer zu erreichen. Deshalb wurden im Rahmen dieser Arbeit einige Anstrengungen zur Entwicklung eines geeigneten Approximationsverfahrens unternommen, das eine stabile und insbesondere effiziente Approximation der Wertfunktion erlaubt und dabei nicht so abhängig von optimal eingestellten Parametern ist.

Das hierbei entwickelte ClusterRL-Verfahren mit sich automatisch an die Lage der Daten im Merkmalsraum anpassenden irregulären Gitterapproximatoren ist ein weiterer wichtiger Beitrag dieser Arbeit. Das ClusterRL-Verfahren ist eine Übertragung und Weiterentwicklung bestehender Ansätze zur Verwendung von Vektorquantisierung, Neural Gas und SOMs im online Reinforcement Lernen auf den batch Fall. Im Gegensatz zu vielen der bestehenden Verfahren is das neue Verfahren vollständig durch theoretische Ergebnisse gedeckt und auch im praxisrelevanten modellfreien Fall nachweislich stabil.

Zwar sind die erzeugten irregulären Gitter wie reguläre Gitter auf die Approximation in Zustandsräumen mit sehr wenigen Dimensionen beschränkt und in dieser Hinsicht anderen, leistungsfähigeren Approximatoren unterlegen. In der Kombination mit einer Technik zur Dimensionsreduktion wie dem tiefen Lernen sind sie dennoch für eine größere Klasse von Problemen auch in der Praxis interessant, zumal sie einfach anzuwenden sind und den Vorteil der gesicherten Stabilität mitbringen.

1.9 Vorgehen und Aufbau der Arbeit

Abbildung 1.3 zeigt schematisch den Aufbau dieser Arbeit. Nach einer kurzen Übersicht über die Grundlagen in Kapitel 2 folgt in Kapitel 3 die Beschreibung des neuen DFQ-Verfahrens zum Reinforcement Lernen in visuomotorischen Problemstellungen. Es werden insbesondere Methoden diskutiert, die ein stabiles und effizientes Lernen auch auf Basis sich verändernder Merkmalsräume ermöglichen. Im Kapitel 4 werden zunächst die theoretischen Eigenschaften von DFQ hinsichtlich Konvergenz und stochastischer Konsistenz diskutiert, um anschließend die Realisierung der beiden zentrale Module des DFQ-Verfahrens zum tiefen Lernen und zur Approximation der Wertfunktion in den Kapiteln 5 und 6 eingehend zu besprechen. Hier spielen auch in Kapitel 4 aus der Theorie heraus formulierte Anforderungen eine wichtige Rolle. So werden in Kapitel 5 die tiefen Lernverfahren hinsichtlich ihrer Tauglichkeit in der geplanten RL-Anwendung untersucht und Strategien für ihren erfolgreichen Einsatz entwickelt. Im Kapitel 6 werden dann die Probleme und Anforderungen an den Funktionsapproximator in DFQ diskutiert. Mit dem ClusterRL Verfahren wird ein neues, stabiles Verfahren eingeführt und evaluiert, das lokale Generalisierung bietet und sich selbständig den Eigenschaften der automatisch erzeugten Merkmalsräume anpasst.

Dabei werden die Algorithmen in den Kapiteln 5 und 6 im Wesentlichen in maßgeschneiderten Simulationen evaluiert, die es anders als reale Systeme ermöglichen, sich auf die wesentlichen Aspekte zu konzentrieren und alle anderen, praktischen Schwierigkeiten auszublenden. In Kapitel 7 wird der vollständige DFQ-Algorithmus auf ein simuliertes, visuomotorisches Lernproblem angewendet und in verschiedenen Varianten ausführlich evaluiert. In der Simulation kommt es ganz notwendigerweise zu Vereinfa-

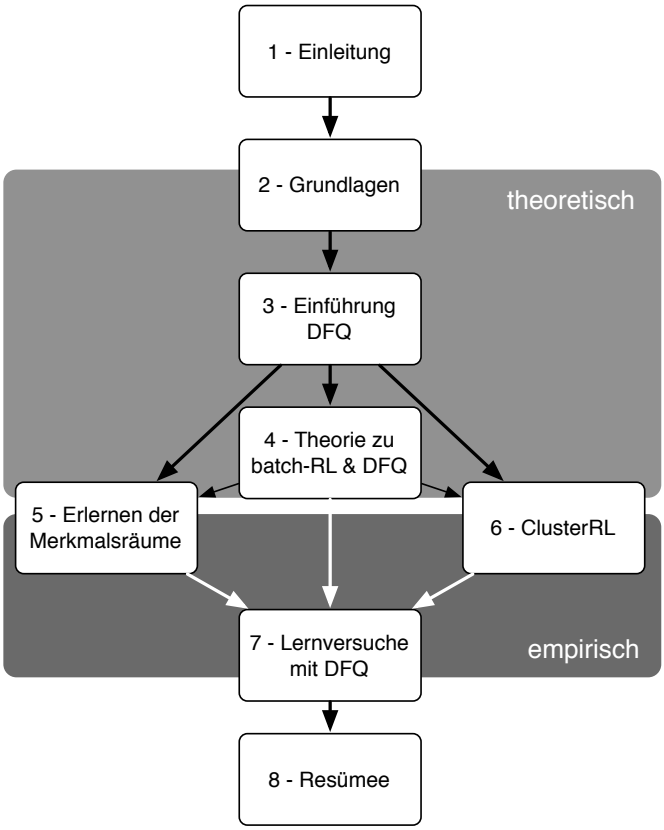


Abbildung 1.3: Übersicht über den Aufbau der Arbeit.

chungen gegenüber den realen Systemen, auch wenn hier anders als bei Gordon, Ernst und Jodogne großer Wert auf die Simulation des unvermeidbaren Bildrauschens und die Verwendung nicht wiederkehrender Bilder – und damit auf aussagekräftige Ergebnisse – gelegt wird. Die getroffenen Vereinfachungen fallen in weiteren Untersuchungen an realen Anwendungen gänzlich weg. Zunächst wird ein realer Bildformationsprozess eingeführt, indem das Benchmarkproblem mit in Echtzeit vom Bildschirm abgefilmten Bildern gelöst wird. Abschließend gelingt es, mit der Carrerabahn ein anspruchsvolles [74], dynamisches System auf Basis von einer Kamera aufgezeichneten Bildern in Echtzeit zu steuern. Nach aktuellem Kenntnisstand ist dies die überhaupt erste erfolgreiche RL-Anwendung ohne maßgeschneiderte Bildverarbeitung, direkt auf den unvorverarbeiteten Bildern einer realen Kamera. Allein schon die vervielfachte Bildgröße mit über 5000 Pixeln ist ein Leistungssprung gegenüber den MNIST-Experimenten auf Bildern mit unter 900 Pixeln.

Diese Anwendungen und die durchgeführten Auswertungen konzentrieren sich auf Problemstellungen, bei denen sich ein zu steuerndes Objekt unter einer stationären Kamera bewegt. Im Vergleich mit den bisher im tiefen Lernen untersuchten Objekter-

kennungsproblemen fügen diese Aufgabenstellungen den zusätzlichen Aspekt der Positionsbestimmung hinzu. In der klassischen Lösung würde dies ein System zur Objektverfolgung erfordern. Im Hinblick auf die Anwendung von DFQ sind diese visuomotrischen Aufgabenstellungen besonders interessant, da viele standardisierte Benchmarkprobleme aus der Literatur in diese Klasse fallen, wenn man sie um eine synthetische Bilderzeugung erweitert oder in der Realität mit einer Kamera implementiert (z.B. Grid-World [142], Mountain-Car [101], inverses Pendel [2]). Wenn diese Problemklasse gelöst wird, ergeben sich unmittelbar reale Anwendungensmöglichkeiten, z.B. in der Fertigungstechnik bei der zielgerichteten Steuerung eines Roboterarms oder anderen Aktuators.

Den Abschluß bildet schließlich Kapitel 8 mit einem Resümee der Arbeit inklusive einer ausführlichen Diskussion und Einordnung des DFQ-Verfahrens.

2

Grundlagen

In diesem Kapitel werden im Bereich des tiefen Lernens und des optimierenden Lernens die in der Arbeit benötigten Grundlagen und Formalismen eingeführt.

2.1 Tiefes Lernen

Die in dieser Arbeit zum Einsatz kommenden tiefen Autoencoder sind eine spezielle Form der künstlichen neuronalen Netze. In der Folge werden zunächst die Grundlagen des Aufbaus und Trainings neuronaler Netze und flacher Autoencodernetze beschrieben, bevor auf das "tiefe Lernen" genannte Training vielschichtiger, "tiefer" Autoencodernetze eingegangen wird. Die später wichtigen Erkenntnisse sind neben der eingeführten Notation die Anordnung der Gewichte zwischen zwei aufeinanderfolgenden Schichten in einer Gewichtsmatrix, die Unterscheidung von online und batch Trainingsverfahren, die mangelnde Ausdruckskraft flacher Autoencoder, die Idee des schichtenweisen Vortrainings tiefer Netze und die Vorzüge RProps gegenüber gewöhnlichen Backpropagations.

2.1.1 Künstliche neuronale Netze

Künstliche neuronale Netze (engl. Artificial Neural Network, ANN) realisieren eine Form des subsymbolischen Lernens und wurden ursprünglich den im Zentralnervensystem von Vertebraten vorgefundenen Nervenzellen und deren Verschaltung nachempfunden [96]. Sie bestehen in der Regel aus künstlichen Neuronen, die eine lokale Berechnung vornehmen, und aus einer Verbindungsstruktur, entlang der die Ergebnisse der Berechnungen in Form von Aktivierungen – oftmals mit einer Verbindungsstärke gewichtet – weitergeleitet werden. Von der Vielzahl der unterschiedlichen Netztypen sind hier die in dieser Arbeit ausschließlich zum Einsatz kommenden mehrschichtigen Perzeptronnetze (engl. Multi Layer Perceptron, MLP) von Bedeutung. Hinsichtlich einer ausführlichen Behandlung alternativer Netztypen und in dieser Arbeit nicht relevanter Eigenschaften und Verfahren sei an dieser Stelle auf die Sekundärliteratur verwiesen [17, 136, 176].

2.1.2 Mehrschichtige Perzeptronnetze

Netzstruktur Ein MLP [141] ist ein künstliches neuronales Netz, bei dem spezielle Perzeptronen [137] $n \in N$ in einem azyklischen Graphen vernetzt sind (s. Abb. 2.1). Jede Verbindung des azyklischen Graphen besitzt ein Gewicht w, wobei w_{ji} das zur Verbindung vom Neuron n_i zum Neuron n_j gehörende Gewicht bezeichnet. Jedes Neuron besitzt einen Schwellwert (Bias) θ und eine differenzierbare, monotone Aktivierungsfunktion f. Um die Ausgabe o_j des Neurons n_j zu bestimmen, wird zunächst die Netzeingabe net_j aus den Ausgaben o_i der Vorgängerneuronen n_i nach $\text{net}_j = \sum_i o_i w_{ji}$ berechnet. Die Aktivierung a_j ergibt sich dann aus $a_j = f_j(\text{net}_j - \theta_j)$. Aus Gründen der technischen Vereinfachung kann der Schwellwert θ_j auch als ein zusätzliches Gewicht w_{j0} von einem speziellen "On-Neuron" n_0, das immer die konstante Aktivierung 1 hat, realisiert werden. In diesem Fall beginnt die Zählung der regulären Neuronen des MLPs bei 1 und für die Netzeingabe folgt $\text{net}_j = w_{j0} + \sum_i o_i w_{ji}$. Die Aktivierung ist dann einfach $a_j = f(\text{net}_j)$, wobei $w_{j0} = -\theta_j$. Aufgrund dieser Realisierungsmöglichkeit als weiteres Gewicht wird im Folgenden, wo nicht unbedingt erforderlich, nicht gesondert auf den Schwellwert eingegangen.

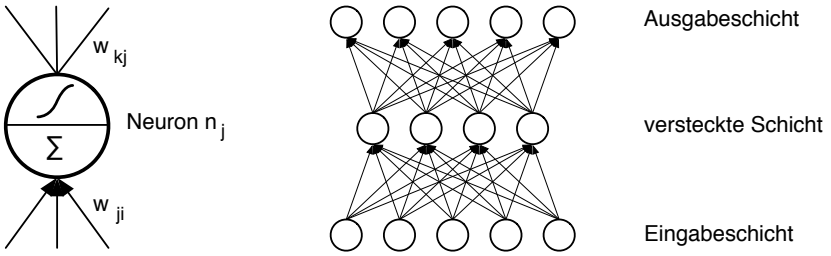


Abbildung 2.1: Darstellung eines mehrschichtigen Perzeptronnetz. Links: Einzelnes Neuron einer versteckten Schicht, das aus der gewichteten Summe der anliegenden Eingangssignale eine nicht-lineare Aktivierung berechnet und an die nachfolgenden Neuronen weiterleitet. Rechts: Anordnung der Neuronen eines dreischichtigen Perzeptronnetzes in einem azyklischen Graphen mit Eingabe-, Ausgabe- und versteckter Schicht.

Die Neuronen $n \in N$ jedes MLPs lassen sich schichtenweise anordnen. Dabei bilden die Neuronen, die keinen Vorgänger besitzen, die Eingabeschicht $I = N_1$ und die Neuronen ohne Nachfolger die Ausgabeschicht $O = N_{h+2}$. Die Mengen I und O sind üblicherweise disjunkt. Für die übrigen Neuronen lässt sich aufgrund der Anordnung in einem azyklischen Graphen ebenfalls eine disjunkte Zerlegung in $h = 0, 1, 2, \ldots$ versteckte Schichten $N_2, ..., N_{h+1}$ mit $\bigcup_{i=1}^{h+2} N_i = N$ und $\bigcap_{i=1}^{h+2} N_i = \emptyset$ finden, so dass jedes Neuron nur ausgehende Verbindungen zu Neuronen nachfolgender Schichten besitzt, also für alle Verbindungen w_{ji} von einem Neuron $n_i \in N_k$ zu einem Neuron $n_j \in N_l$ stets $k < l$ gilt. Aus diesem Grund lassen sich die Neuronen des gesamten Netzes auch so durchnummerieren, dass aus $k < l$ auch $i < j$ folgt. In dieser Weise nummeriert, können

die Verbindungsgewichte w_{ji} in eine $|N| \times |N|$-Matrix (w_{ji}) der Form

$$W = \begin{pmatrix} w_{1,1} & \cdots & w_{1,|N|} \\ \vdots & \ddots & \vdots \\ w_{|N|,1} & \cdots & w_{|N|,|N|} \end{pmatrix}$$

eingetragen werden, wobei nicht vorhandene Verbindungen den Eintrag $w_{ji} = 0$ erhalten. Im Falle der MLPs ist die Gewichtsmatrix W eine strikte untere Dreiecksmatrix.

Sind die Netze nur schichtenweise verbunden, d.h. es gibt nur Verbindungen zwischen Neuronen aufeinander folgender Schichten und keine schichtenübergreifenden "Shortcut-Verbindungen", können die Gewichte w_{ji} alternativ in mehreren kleineren und dichter besetzten Matrizen $W^1, W^2, \ldots, W^{h+1}$ gespeichert werden, wobei die Matrix W^k die Gewichte der Verbindungen zwischen Schicht k und $k+1$ umfasst. Die Nummerierung der Gewichte, Neuronen, Aktivierungen, etc. erfolgt dann innerhalb der einzelnen Schichten, wobei n_i^k das i-te Neuron der Schicht k und w_{ji}^k die Verbindung vom i-ten Neuron der Schicht k zum j-ten Neuron der Schicht $k+1$ bezeichnen. W^k ist eine $|N_{k+1}| \times |N_k|$ - Matrix. Bei schichtenweise vollverbundenen Netzen gilt darüber hinaus für alle Einträge w_{ji}^k der Matrix W^k immer $w_{ji}^k \neq 0$. Im weiteren Verlauf wird von dieser Art der Nummerierung ausgegangen und das Superscript k zur Kennzeichnung der Schicht nur dort explizit angegeben, wo es nicht aus dem Kontext erschlossen werden kann.

Vorwärtspropagieren Daten werden durch das Netz propagiert, indem zunächst die Aktivierungen der Eingabeneuronen auf die Eingabe x gesetzt werden, also $a_j^1 = x_j$. Aktivierungen werden nun entlang des Verbindungsgraphen bis hin zur Ausgabeschicht propagiert, wofür in der schichtenbezogenen Matrixschreibweise gilt:

$$\begin{pmatrix} net_1^{k+1} \\ \vdots \\ net_{|N_{k+1}|}^{k+1} \end{pmatrix} = \begin{pmatrix} w_{1,1}^k & \cdots & w_{1,|N_k|}^k \\ \vdots & \ddots & \vdots \\ w_{|N_{k+1}|,1}^k & \cdots & w_{|N_{k+1}|,|N_k|}^k \end{pmatrix} \cdot \begin{pmatrix} o_1^k \\ \vdots \\ o_{|N_k|}^k \end{pmatrix} .$$

Die Aktivierungen der Ausgabeneuronen bilden dann die Netzausgabe o mit $o_j = a_j^{h+2}$. Als Aktivierungsfunktion kommt in dieser Arbeit ausschließlich die sigmoide Funktion

$$f_{sgd}(x) = \frac{1}{1+e^{-x}} \qquad f'_{sgd}(x) = f_{sgd}(x)(1 - f_{sgd}(x))$$

zum Einsatz.

Lernen Trainiert werden die MLPs auf einem Satz von Trainingspattern $p = (x; y) \in \mathcal{P}$ mit Eingabevektoren x und Zielvektoren y mit Hilfe eines Gradientenabstiegsverfahrens. Dabei werden die Verbindungsgewichte über einen Abstieg auf dem Gradienten $\nabla E(W)$ so angepasst, dass ein Fehler $E(W) = \sum E_p(W)$ als Summe der patternbezogenen Fehler $E_p(W)$ zwischen den unter den Gewichten W erzeugten tatsächlichen

Netzausgaben und den vorgegebenen Zielwerten minimiert wird. Der gebräuchlichste und hier meist verwendete Fehler ist die Summe der quadratischen Fehler (SSE) auf den Trainingsbeispielen

$$E = \frac{1}{2} \sum_{p=1}^{|P|} \sum_{i=1}^{|O|} (y_i^p - o_i^p)^2 \ .$$

Die Berechnung der Werte der partiellen Ableitungen $\frac{\partial E}{\partial w_{ji}}$ der Fehlerfunktion E nach den einzelnen Gewichten w_{ji} kann mit Hilfe des Backpropagation-Algorithmus [140, 172] berechnet werden. Im Backpropagation-Algorithmus wird die Gewichtsänderung dann in Erweiterung der Delta-Regel [173] in Richtung des negativen Gradienten der Fehlerfunktion vorgenommen

$$\Delta W = -\eta \nabla E(W) \ ,$$

wobei sich für das einzelne Gewicht w_{ji} die Änderung

$$\Delta w_{ji} = -\eta \frac{\partial}{\partial w_{ji}} E(W)$$

ergibt. η ist dabei eine über die Zeit kleiner werdende Lernrate.

Statt wie in dieser batch Variante des Backpropagation-Algorithmus zunächst den Gradienten $\nabla E(W)$ auf allen Trainingsbeispielen (dem gesamten "Batch") zu bestimmen, werden im online Backpropagation die Gewichtsänderungen sofort nach jeder Berechnung des Teilfehlers für eines der Trainingsbeispiele nach

$$\Delta w_{ji} = -\eta \frac{\partial}{\partial w_{j,i}} E_p(W)$$

vorgenommen. Weil eine einzelne solche Änderung dann aber von der Richtung des kleiner werdenden Gesamtfehlers $-\nabla E(W)$ abweichen und auch zu vorübergehenden Verschlechterungen führen kann, wird dieses Verfahren als stochastischer Gradientenabstieg bezeichnet. Ein Zwischending zwischen reinem online und reinem batch Verfahren ist das Training auf sogenannten Mini-Batches, bei dem der Gradient $\nabla E(W)$ vor jeder Gewichtsänderung auf einer größeren Teilmenge der Trainingsdaten approximiert wird.

Regularisierung Einerseits wird versucht, den Fehler auf den Trainingsdaten während des Gradientenabstiegs so weit wie möglich zu verkleinern, andererseits ist man eigentlich an der Generalisierungsfähigkeit des Netzes interessiert, d.h. es sollen möglichst korrekte Ausgaben für neue Eingaben erzeugt werden. Typischerweise ist bei einem zu langen Training ein Überspezialisieren der Gewichte auf die Trainingsdaten zu beobachten; der Fehler auf nicht im Training verwendeten Daten steigt schon wieder an, während der Trainingsfehler noch kleiner wird. Um dies zu verhindern, kommen verschiedene Regularisierungstechniken (Early Stopping, Weight Decay, Pruning) zum Einsatz.

Resilient Propagation In der Praxis hat sich vielfach die RProp-Aktualisierungsregel [131] bewährt [63, 127, 131]. RProp ist ein reines batch Verfahren, das auf einer Berechnung des Gradienten der Fehlerfunktion wie bei Backpropagation basiert, dann aber nicht die Stärke des Gradienten, sondern nur seine Richtung zur Bestimmung der Gewichtsänderung verwendet. Die Schrittweite wird durch eine Heuristik an jedem Gewicht einzeln bestimmt: Zeigt der Gradient in aufeinander folgenden Schritten in die gleiche Richtung, wird die Schrittweite erhöht, besitzt der Gradient ein anderes Vorzeichen, wird die Schrittweite verkleinert. Es kommt mit vergleichsweise wenigen Parametern aus und hat sich bezüglich der Werte dieser Parameter als äußerst robust erwiesen [131]. In Bezug auf das hier verfolgte Training tiefer Netze erscheint RProp zusätzlich interessant, da es auch weniger anfällig für kleiner werdende Gradienten ist. Es werden nur die Vorzeichen, nicht aber die Beträge betrachtet. Hinsichtlich der Regularisierung lässt sich RProp genau wie Standard-Backpropagation mit Weight Decay und Early Stopping kombinieren.

Training vielschichtiger Netze Ungelöst war bisher das Training tiefer Netze mit mehr als ein oder zwei versteckten Schichten. Nach allgemeiner Erfahrung lassen sich in diesen Netzen mit Hilfe von Backpropagation keine guten Ergebnisse erzielen [12]. Netze neigen entweder zum Überspezialisieren oder aber zum Erlernen der Durchschnittsausgabe, ohne "Feinheiten" erfassen zu können. Ein Grund ist, dass der Gradient beim Zurückpropagieren durch viele Schichten oftmals entweder sehr klein oder sehr groß wird, so dass jeglicher Informationsgehalt verloren geht [12]. Dieses Phänomen ist als Problem des "diminishing gradient" bekannt. Mit dem "tiefen Lernen" stehen nun neue Techniken zum Training solcher Netze zur Verfügung. Sie werden nach einer kurzen Diskussion der in diesem Zusammenhang relevanten Autoencodernetze vorgestellt.

2.1.3 Autoencoder

Ein (flacher) Autoencoder (oder auch "Autoassociator", siehe Abbildung 2.2) ist ein spezielles, dreischichtiges Perzeptronnetzwerk, das schichtenweise vernetzt ist, in der Eingabe- und Ausgabeschicht gleich viele Neuronen besitzt und zu Zwecken der Datenkomprimierung bzw. Merkmalsextraktion eingesetzt werden kann. Typischerweise besitzen Autoencodernetze einen "Flaschenhals", d.h. es befinden sich in der einzigen versteckten Schicht weniger Neuronen als in den äußeren Schichten. In der Regel sind die Schichten voll vernetzt, aber auch symmetrische (achsensymmetrisch in Bezug auf die innere Schicht) und vollkommen unsymmetrische Vernetzungstrukturen zwischen den Schichten sind möglich.

Ein Autoencodernetz lässt sich in zwei logische Teilnetze zerlegen; Eingabe- und versteckte Schicht bilden einen Encoder, während versteckte Schicht und Ausgabeschicht dem zugehörigen Decoder entsprechen. Der Encoder berechnet dabei eine Kodierung z der Eingabedaten mit $z = \mathrm{ENC}(x; W^{ENC})$ und der Decoder die Inverse $\hat{x} = \mathrm{DEC}(z; W^{DEC})$. Im Autoencoder realisieren diese beiden Teilnetze dann die Abbildung $\hat{x} = \mathrm{DEC}(\mathrm{ENC}(\mathbf{x}; W^{enc}); W^{dec})$. Durch die Kombination des Encoders und Decoders im Autoencoder wird ein Training der Gewichtsvektoren auf die Identitätsfunktion

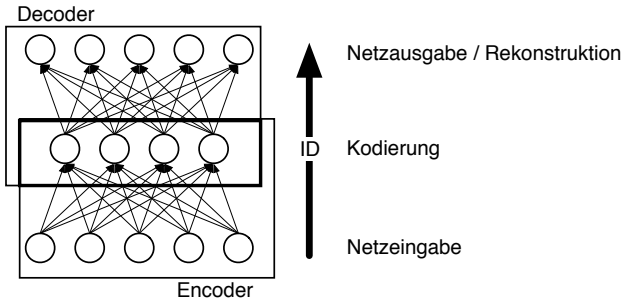


Abbildung 2.2: Autoencoder mit fünf Eingabeneuronen und vier Neuronen in der versteckten Schicht zur Kodierung von fünfdimensionalen Eingaben in vierdimensionalen Vektoren. Trainiert werden diese Netze auf die möglichst genaue Rekonstruktion der angelegten Eingaben in der Ausgabeschicht.

$id(x) = x$ ermöglicht, indem ein Gradientenabstieg auf dem Rekonstruktionsfehler, im Falle von reellwertigen, normierten Daten beispielsweise

$$\mathrm{RE} = \frac{1}{2} \sum_{p \in \mathcal{P}} \sum_{i=1}^{|N_{h+1}|} (\hat{x}_i^p - x_i^p)^2 \ ,$$

erfolgt. Hierdurch wird in der gemeinsamen versteckten Schicht – auch Kodierungsschicht genannt – eine Kodierung z der Eingabedaten erlernt, die trotz der geringeren Anzahl der Dimensionen möglichst viele Informationen erhält und so eine gute Rekonstruktion der angelegten Daten in der Ausgabeschicht erlaubt.

Aufgrund der Verwendung der Identitätsfunktion und der Beschränkung auf die Eingabevektoren wird das Training von Autoencodernetzen oftmals dem unüberwachten, selbstorganisierenden Lernen zugeordnet, obwohl natürlich streng genommen lediglich die klassischen überwachten Lernverfahren zum Einsatz kommen.

Die Idee der einschichtigen Autoencodernetze zur Lösung des Encoder-Problems [1] wurde schon in der Frühphase der MLPs als eine Anwendung des Backpropagation Algorithmus für MLPs vorgeschlagen [139] und auch schon früh in der Linguistik [37] und Bildverarbeitung [27] eingesetzt. Die Ausdrucksstärke dieser flachen Autoencoder-Netze ist aber beschränkt; in [19] wurde nachgewiesen, dass die hinsichtlich des Rekonstruktionsfehlers optimalen Gewichte eines Autoencodernetzes mit linearer Aktivierung in der Ausgabeschicht durch eine (lineare) Singular Value Decomposition (SVD) direkt berechnet werden können— auch trotz nichtlinearer Aktivierungen in der versteckten Schicht. Somit können nur Merkmale als Linearkombinationen der Eingaben gefunden werden, die prinzipiell dem Ergebnis einer Hauptkomponentenanalyse gleichen [6]. Weil zufällig initialisierte Autoencoder beim Gradientenabstieg auf dem Rekonstruktionsfehler oftmals in lokalen Minima stecken bleiben, konnten zudem in den in [19] durchgeführten Versuchen per SVD allgemein bessere Rekonstruktionsergebnisse als mit den Autoenco-

dern erzielt werden. Wohl auch aufgrund dieser Ergebnisse spielen Autoencoder in der Praxis bisher eine eher untergeordnete Rolle.

2.1.4 Training tiefer Autoencoder

Durch die Hinzunahme weiterer versteckter Schichten vor und nach der Kodierungsschicht könnte die Ausdrucksstärke von Autoencodern erhöht werden. Dadurch würde auch die automatische Extraktion von komplexeren, nichtlinearen Merkmalen möglich, was diese "tiefen Autoencoder" (siehe Abb. 2.3) für eine Vielzahl weiterer Anwendungen interessant machen würde.

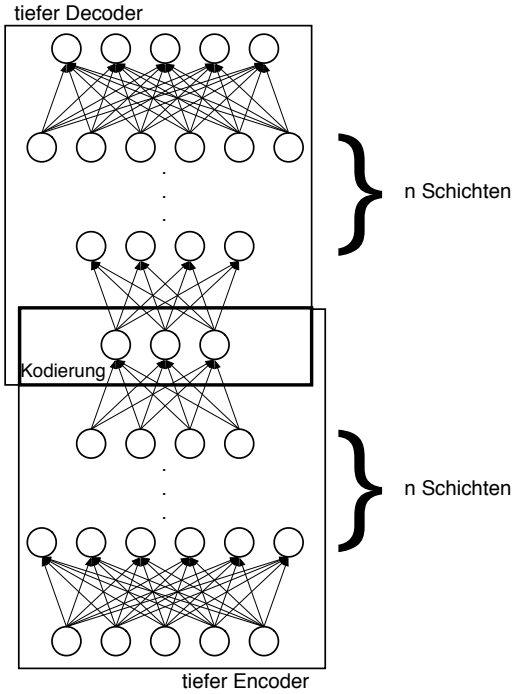


Abbildung 2.3: Tiefer Autoencoder mit zusätzlichen versteckten Schichten zur Extraktion von nichtlinearen Merkmalen vor und nach der Kodierungsschicht. Der Aufbau des Netzes bleibt wie beim dreischichtigen Autoencoder symmetrisch.

Lange Zeit war das Training solcher tiefen Architekturen aber im Gegensatz zu den flachen Netzen nicht effektiv möglich. Zuletzt wurden nun verschiedene neue Techniken zum Training solcher tiefen Architekturen vorgestellt und unter dem Schlagwort "tiefes Lernen" (engl. Deep Learning, DL) beworben. Tiefe Autoencoder sind dabei nur ein Beispiel für den prinzipiellen Nutzen tiefer Architekturen. Bengio liefert mit [12] einen zugänglichen Überblick über dieses noch junge, aber schnell wachsende Gebiet.

Von besonderem Interesse ist an dieser Stelle ein von Hinton entwickeltes und in

Science vorgestelltes Verfahren zum Training tiefer Autoencodernetze, die zur nichtlinearen Reduktion der Dimensionalität von hochdimensionalen Eingabedaten eingesetzt werden sollen [61]. Das verfolgte Ziel ist das Erlernen einer möglichst informationserhaltenden Kodierung d_I-dimensionaler Daten in d_C-dimensionalen Merkmalsvektoren. Die Kodierung kann nun durch ein tiefes Encodernetz mit mehreren versteckten Schichten H_i mit $i = 1, \ldots, h$ zwischen der Eingabeschicht I und der Ausgabeschicht O bzw. Kodierungsschicht $C = O$ realisiert werden. Die Anzahl der versteckten Schichten h und die Menge d_{H_i} der Neuronen in den einzelnen versteckten Schichten muss dabei vorgegeben werden. Es macht dabei durchaus Sinn, die Breite der Repräsentation zunächst zu vergößern und in der ersten versteckten Schicht mehr Neuronen als in der Eingabeschicht vorzusehen, um dann die Dimensionalität schrittweise bis zur Kodierungsschicht zu reduzieren. Statt diesen Encoder nun aber sofort mit dem passenden Decodernetz gemeinsam in einem tiefen Autoencoder auf den Daten zu trainieren, führt Hinton eine vorgeschaltete Initialisierungsphase ein, in der die Gewichte des tiefen Autoencoders mit einem besonderen Verfahren schichtenweise von außen nach innen vortrainiert und in die Nähe eines lokalen Optimums geführt werden. Anders als mit zufällig initialisierten Gewichten führt der Gradientenabstieg auf den in solcher Art vortrainierten Gewichten zum Erfolg.

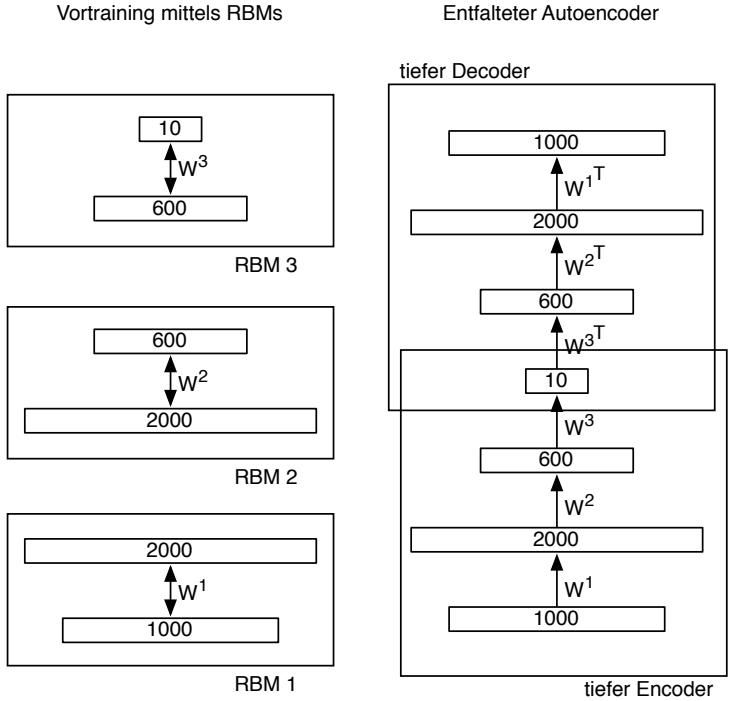


Abbildung 2.4: Schematische Darstellung des Vortrainings tiefer Netze mittels mehrerer RBMs. Zeichnung in Anlehnung an [61].

Das von Hinton vorgeschlagene Vorgehen in der Vortrainingsphase ist in Abbildung

2.4 grafisch dargestellt. Begonnen wird mit dem Vortraining der Gewichte in den äußeren Schichten des tiefen Autoencoders. Dazu wird ein flaches Netz konstruiert, dessen erste Schicht der Eingabeschicht N_1 des tiefen Autoencoders und dessen zweite Schicht der ersten versteckten Schicht N_2 des tiefen Netzes entsprechen.

Während Hinton für den tiefen Autoencoder selbst MLPs verwendet, kommen im Vortraining mit der Restricted Boltzmann Machine (RBM) [43, 152] rekurrente Netze mit stochastischer, binärer Aktivierung mit $P(a_i = 1|net_i) = \frac{1}{1+e^{net_i}}$ zum Einsatz [60, 61]. Der Aufbau der RBMs ist in Abbildung 2.5 schematisch dargestellt. Die Gewichte W^1 der ersten RBM werden mit Hilfe des Contrastive Divergence Verfahrens [59] auf den Eingabedaten trainiert. Nach Abschluss des Trainings dienen die Aktivierungen der versteckten Schicht der RBM als Eingabedaten für das Training der nächsten RBM, deren Eingabeschicht die gleiche Größe der versteckten Schicht der Vorgänger-RBM besitzt, während die versteckte Schicht der nächsten Schicht im Encoder entspricht. Dieses Vorgehen wird wiederholt, bis man an der Kodierungsschicht angelangt ist.

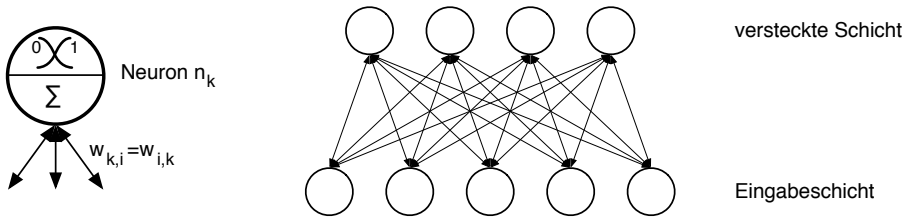


Abbildung 2.5: Schematische Darstellung der Restricted Boltzmann Machine, bei der im Gegensatz zu der nicht eingeschränkten Boltzmann Maschine keine Verbindungen zwischen Neuronen derselben Schicht erlaubt sind. Bei der RBM ist die Eingabeschicht gleichzeitig die Ausgabeschicht des Netzwerkes.

Nach Abschluss dieses schichtenweisen Vortrainings können die in den stochastischen RBMs trainierten Gewichte als Grundlage für die Initialisierung eines deterministischen MLPs dienen. Die einzelnen RBMs werden, wie in Abbildung 2.4 dargestellt, zu einem tiefen Autoencodernetz auseinandergefaltet. Die in den RBMs initialisierten Gewichte befinden sich, anders als zufällig initialisierte Gewichte, nun in der Nähe eines "guten" lokalen Minimums und können mit Hilfe gewöhnlicher Gradientenabstiegsverfahren auf dem Rekonstruktionsfehler weiter trainiert und verfeinert werden [61].

In einer ausführlichen Diskussion dieser Verfahren weist Bengio darauf hin, dass die Idee des schichtenweisen Vortrainings nicht an die Verwendung von RBMs gebunden ist, sondern prinzipiell auch mit Hilfe von gewöhnlichen flachen Autoencodern angewendet werden kann (siehe [12, Abschnitt 9]). An diese Feststellung wird in Kapitel 5 angeknüpft, wo dann die in dieser Arbeit eingesetzte Variante des Vortrainings eingehend erläutert wird.

2.2 Reinforcement Lernen

Der als Reinforcement Lernen bekannte Teilbereich des maschinellen Lernens wird nach Sutton und Barto [156] durch das Reinforcement-Lernproblem definiert. Dieses Problem ist durch die in Abbildung 2.6 dargestellte Agent-Umgebungs-Schleife gekennzeichnet, in der ein Agent mit einer Umgebung interagiert und versucht, sich dabei über die Auswahl geeigneter Aktionen möglichst optimal zu verhalten. "Optimal" heißt hierbei, ab dem Zeitpunkt $t = 1, 2, \ldots$ die zukünftige, akkumulierte Belohnung (engl. reward)

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

zu maximieren, wobei γ ein Faktor mit $0 \leq \gamma \leq 1$ zur Gewichtung weiter in der Zukunft liegender Belohnungen r_{t+k+1} ist.

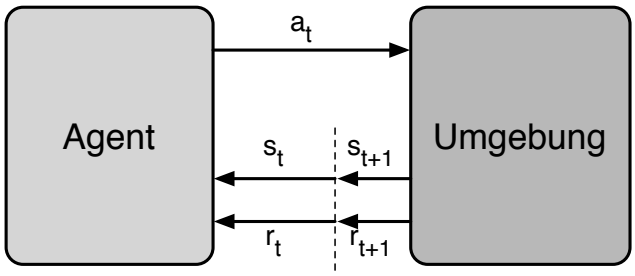


Abbildung 2.6: Zeitdiskrete Agent-Umgebungsschleife im Reinforcement Lernen. Der Agent nimmt in jedem Zeitschritt den Zustand s_t der Umgebung war und reagiert mit der Auswahl einer Aktion a_t. Zusätzlich erhält er in jedem Schritt ein Belohnungssignal r_t.

Zur Maximierung dieser Belohnung muss vom Agenten ein sequentielles Entscheidungsproblem gelöst werden, bei dem die in einer Situation richtige Ausgabe (Aktion) nicht nur von der sofortigen Belohnung r_{t+1} (engl. immediate reward) abhängt, sondern auch Auswirkungen auf zukünftig zu erwartende Belohnungen berücksichtigt werden müssen. Dieses Problem der Zuordnung zukünftiger Erfolge oder Misserfolge bzw. allgemein "verzögerter Belohnungen" (engl. delayed reward) zu einzelnen in der Sequenz getroffenen Entscheidungen wird als Credit-Assignment Problem [99, 146] bezeichnet. Die allgemeinsten Methoden zur Lösung von RL-Problemen sind die auf dem dynamischen Programmieren [10] fundierenden wertfunktionsbasierten Ansätze.

2.2.1 Markovsche Entscheidungsprozesse

Klassisch werden Reinforcement Lernprobleme als Markovsche Entscheidungsprozesse (engl. Markov Decision Process, MDP) modelliert. Ein MDP ist ein Viertupel MDP =

(S, A, T, R) mit einer Menge von Zuständen S, einer Menge von Aktionen A, zeitdiskreten Zustandsübergängen von einem Ausgangszustand $s \in S$ zu einem Nachfolgezustand $s' \in S$ nach

$$T : S \times A \times S \mapsto \mathbb{R} : (s, a, s') \mapsto P(s_{t+1} = s' | s_t = s, a_t = a)$$

und einer Belohnungsfunktion

$$R : S \times A \times S \mapsto \mathbb{R} : (s, a, s') \mapsto E\left[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\right] \ ,$$

die den Erwartungswert der sofortigen Belohnung r_{t+1}beim Übergang von Zustand s in den Nachfolgezustand s' unter Wahl der Aktion $a \in A$ angibt.

Eine wichtige Eigenschaft dieser besonderen Art stochastischer Prozesse ist die "Gedächtnislosigkeit": Bei einem MDP hängt der angenommene Folgezustand lediglich vom aktuellen Zustand und der aktuell gewählten Aktion, nicht aber von der Historie der vorhergehenden Zustände und Aktionen ab. Es gilt also insbesondere für die Übergangswahrscheinlichkeiten die Markov-Eigenschaft für Zustandsübergänge und Belohnungen:

$$P(s_{t+1} = s' | s_t, a_t) = P(s_{t+1} = s' | s_t, a_t, s_{t-1}, a_{t-1}, \ldots) \quad (2.1)$$

$$E\left[r_{t+1} | s_t, a_t, s_{t+1}\right] = E\left[r_{t+1} | s_{t+1}, a_t, s_t, a_{t-1}, s_{t-1}, \ldots\right] \quad . \quad (2.2)$$

Das Ziel ist es, eine stationäre Strategie (engl. policy) $\pi : S \mapsto A$ zu finden, die für jeden Zustand $s \in S$ genau die Aktion $a \in A$ auswählt, die den Erwartungswert $E[R_t | s_t = s]$ der akkumulierten, zukünftigen Belohnung R_t maximiert.

2.2.2 Diskontierte und nicht diskontierte Problemstellungen

Im Rahmen dieser Arbeit werden ausschließlich Probleme mit unendlichem Horizont betrachtet, bei denen es sich entweder um diskontierte Probleme mit $0 \leq \gamma < 1$ oder um nicht diskontierte kürzester Pfad Probleme mit $\gamma = 1$ und mindestens einem absorbierenden Terminalzustand in $S_+ \subset S$ handelt. Einmal erreicht, kann ein absorbierender Terminalzustand nicht mehr verlassen werden und ermöglicht im weiteren Verlauf keine weiteren Belohnungen, also $E[R_t | s_t = s] = 0 \quad \forall s \in S_+$, während alle anderen Zustandsübergänge in der Regel identische Belohnungen (oft auch in Form von Kosten c als negative Belohnungen mit $c = -r$ notiert) z.B. in Höhe von $r = -1$ verursachen. In solchen Problemstellungen kann der Agent die akkumulierte Belohnung maximieren (bzw. die Kosten minimieren), indem er sich zeitoptimal verhält, d.h. sich möglichst schnell in einen der absorbierenden Zielzustände begibt. Damit in solch einer Formulierung die zu erwartenden Kosten von jedem Ausgangszustand beschränkt bleiben [16], ist es notwendig, dass von jedem Zustand $s \in S$ mindestens einer der Terminalzustände in S_+ erreicht werden kann. Neben den positiven Terminalzuständen S_+ (Zielzustände) können in solchen kürzester Pfad Problemen auch für den Agenten negative Terminalzustände S_- (Fehlerzustände) eingeführt werden, die es aufgrund sehr hoher sofortiger Kosten beim Übergang von einem Zustand $s \in S \setminus S_-$ in einen Fehlerzustand $s' \in S_-$ zu vermeiden gilt.

2.2.3 Optimale Wertfunktion und optimale Strategie

Eine zentrale Bedeutung kommt im klassischen RL der optimalen Zustandswertfunktion

$$\begin{aligned} V^*(s) &= E_{\pi^*}[R_t|s_t = s] \\ &= E_{\pi^*}\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2}|s_t = s\right] \\ &= \sum_{s' \in S} T(s, \pi^*(s), s') \left[R(s, \pi^*(s), s') + \gamma V^*(s')\right] \end{aligned} \tag{2.3}$$

zu. Sie gibt für jeden möglichen Zustand s die zu erwartende zukünftige Belohnung an, wenn ausgehend von diesem Zustand in Zukunft die optimale Strategie π^* befolgt wird. Auf gleiche Weise können die Zustandswerte $V^\pi(s)$ unter Verfolgung jeder beliebigen anderen Strategie π berechnet werden. Dazu wird in der Gleichung 2.4, der Bellman-Gleichung für V^π, nicht die optimale Aktion $a_t = \pi^*(s)$, sondern eine von der zu bewertenden Strategie π ausgewählte Aktion $a_t = \pi(s)$ verwendet:

$$\begin{aligned} V^\pi(s) &= E_\pi\left[R_t|s_t = s\right] \\ &= \sum_{s' \in S} T(s, \pi(s), s') \left(R(s, \pi(s), s') + \gamma V^\pi(s')\right) \ . \end{aligned} \tag{2.4}$$

Die Funktion V^π ist als eindeutige Lösung der Bellman-Gleichung ein Maß für die Güte der Strategie π, das zur Strukturierung der Suche nach der optimalen Strategie herangezogen werden kann [156]. Eine Zustandswertfunktion ist dabei als eindeutige Lösung des Bellmanschen-Optimalitätskriteriums – nach einem zentralen Ergebnis des dynamischen Programmierens – genau dann optimal, wenn gilt:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma V^*(s')\right) \quad \forall s \in S \ . \tag{2.5}$$

Ist die optimale Wertfunktion bekannt, kann leicht eine optimale Strategie π^* abgeleitet werden, indem die Zustandswertfunktion "gierig" (engl. greedy) ausgewertet wird:

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma V^*(s')\right) \ . \tag{2.6}$$

2.2.4 Wertiteration

Aus der Bellman-Gleichung (2.4) und dem Optimalitätskriterium (2.5) lassen sich direkt Aktualisierungsvorschriften für verschiedene Algorithmen zum Finden der optimalen Wertfunktion ableiten. Im synchronen Wertiterationsverfahren (engl. Value Iteration) wird ausgehend von einer beliebigen initialen Wertfunktion V^0 in jeder Iteration auf jeden Zustand $s \in S$ die Aktualisierungsvorschrift

$$V^{i+1}(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma V^i(s')\right) \tag{2.7}$$

angewendet, bis sich die Wertfunktion nach einer vollständigen Iteration auf keinem der Zustände $s \in S$ mehr ändert bzw. bis die (maximale) Änderung (engl. Bellman-residual) unterhalb einer vorgegebenen Schranke liegt. Sofern sie existiert, ist die optimale Wertfunktion V^* der Fixpunkt, zu dem die Folge (V^i) der Wertfunktionen in Maximum-Norm mit $||V^i - V^*||_\infty = \max_{s \in S} |V^i(s) - V^*(s)|$ konvergiert [14, 122]. In der Gauß-Seidel Variante dieses Algorithmus wird die Aktualisierungsvorschrift asynchron, d.h. in einer beliebigen Reihenfolge (s_k), auf die Werte $V^i(s_k)$ der Zustände s_k angewendet. Auch hier kann Konvergenz unter der Bedingung sichergestellt werden, dass jeder der Zustände $s \in S$ unendlich oft in der Folge (s_k) vorkommt [16].

Statt die Aktualisierungsregel (2.7) immer auszuschreiben, ist auch folgende Notation gebräuchlich

$$V^{i+1} \quad = \quad HV^i \tag{2.8}$$

oder noch kürzer

$$V \quad = \quad HV \ , \tag{2.9}$$

wobei der Operator H die Abbildung auf Funktionen ist, der die Wertfunktion V^i entsprechend der Vorschrift (2.7) zur Funktion V^{i+1} überführt.

2.2.5 Q-Lernen

Ein Nachteil des Wertiterationsverfahrens ist, dass Gleichung (2.7) genau wie die Ableitung der Strategie (2.6) auf der Kenntnis der Systemdynamik in Form des Übergangsmodells T, R beruht. Ist das Übergangsmodell, wie es in dieser Arbeit durchweg der Fall ist, vorab nicht bekannt, wird ein modellfreier Ansatz wie zum Beispiel Watkins Q-Lernen [169] benötigt. Die zentrale Idee dieser wohl bekanntesten modellfreien Temporal Difference–Methode (Übersicht siehe z.B. [156, Kapitel 6]) ist es, an Stelle der Werte von Zuständen die Werte von Zustands-Aktionspaaren in der sogenannten Q-Funktion

$$\begin{aligned} Q^\pi(s,a) &= E\left[r_{t+1} + \gamma Q^\pi\left(s_{t+1}, \pi(s_{t+1})\right) | s_t = s, a_t = a\right] && (2.10)\\ &= \sum_{s' \in S} T(s,a,s') \left(R(s,a,s') + \gamma Q^\pi\left(s', \pi(s')\right)\right) && (2.11) \end{aligned}$$

zu bestimmen. Für die gesuchte optimale Q-Funktion gilt analog

$$\begin{aligned} Q^*(s,a) &= E\left[r_{t+1} + \gamma Q^*\left(s_{t+1}, \pi^*(s_{t+1})\right) | s_t = s, a_t = a\right] && (2.12)\\ &= \sum_{s' \in S} T(s,a,s') \left(R(s,a,s') + \gamma\ Q^*\left(s', \pi^*(s_{t+1})\right)\right) && (2.13)\\ &= \sum_{s' \in S} T(s,a,s') \left(R(s,a,s') + \gamma \max_{a' \in A} Q^*\left(s', a'\right)\right) && (2.14) \end{aligned}$$

Gleichung (2.14) folgt direkt aus (2.13) durch einsetzten der Definition der optimalen Strategie π^* auf Basis der gierigen Auswertung der optimalen Q-Funktion:

$$\pi^*(s) := \arg\max_{a \in A} Q^*(s,a) \ . \tag{2.15}$$

Statt nun in einer abgeleiteten Lernregel den Erwartungswert in (2.10) bzw. (2.12) mit Hilfe des Übergangsmodells wie in der Wertiteration (2.7) für jeden Ausgangszustand direkt zu berechnen, wird er beim Q-Lernen mit Hilfe stochastischer Approximation [135] über die Auswertung wiederholter, stochastischer Zustandsübergänge vom gleichen, gemeinsamen Ausgangszustand s bestimmt. Für einen beobachteten Übergang (s, a, r, s') wird dazu ausgehend von einer initialen Schätzung Q^0 der optimalen Q-Funktion folgende Aktualisierung vollzogen:

$$Q^{i+1}(s,a) \leftarrow (1-\alpha)Q^i(s,a) + \alpha(r + \gamma \max_{a' \in A} Q^i(s',a'))\,. \tag{2.16}$$

In einem stochastischen System ist für die korrekte implizite Schätzung der Übergangswahrscheinlichkeiten und der damit verbundenen Q-Werte entscheidend, dass die Übergänge (s, a, r, s') entsprechend den unbekannten, T und R zugrunde liegenden Verteilungen gezogen werden. Dies kann leicht sichergestellt werden, indem die Übergänge in der direkten Interaktion mit dem System als Realisierung der (unbekannten) Verteilungen gewonnen werden. Wird zudem jeder einzelne Zustand immer wieder besucht und wird die Lernrate α in geeigneter Weise über die Zeit abgesenkt, konvergiert die Folge (Q^i) für endliche Zustandsmengen S gegen die optimale Q-Funktion Q^* [170].

Der Vorteil dieses Lernverfahrens gegenüber der Wertiteration ist, dass das Übergangsmodell T weder für die Ableitung einer stationären, optimalen Strategie nach (2.15) von der optimalen Q-Funktion Q^* noch für den Lernschritt (2.16) benötigt wird.

2.2.6 Approximatives Reinforcement Lernen

Während die Wertfunktionen für endliche Zustandsräume prinzipiell exakt repräsentiert werden können, zum Beispiel durch Speichern der Q-Werte in einer Tabelle, ist im Falle unendlicher (kontinuierlicher) Zustandsräume, zum Beispiel $S = \mathbb{R}^n$, allein schon unter praktischen Gesichtspunkten nur eine approximative Näherung möglich, da nicht alle Zustände immer wieder besucht und getrennt voneinander aktualisiert werden können. Hier ist der Einsatz von Funktionsapproximatoren nötig, die es ermöglichen, von den wenigen tatsächlich gewonnenen Übergängen auch auf die Q-Werte anderer, nicht besuchter Zustände zu schließen.

Historisch gesehen wurden in verschiedenen approximativen Lernansätzen Aktualisierungsregeln zunächst direkt von den Lernregeln für endliche Zustandsräume abgeleitet und an die eingesetzten Funktionsapproximatoren angepasst. Eine approximative Version des Q-Lernens kann die Aktualisierungsvorschrift (2.16) zum Beispiel mit Hilfe einer in einem neuronalen Netz gespeicherten Approximation $\hat{Q}^i$ realisieren. Die Gewichte des MLPs werden in diesem Fall für einen beobachteten Übergang (s, a, r, s') in Abhängigkeit des Gradienten der Differenz zwischen alter und neuer Schätzung mit der individuellen Änderung

$$\Delta w_{ji} \;=\; \alpha \frac{\partial}{\partial w_{ji}} \frac{1}{2} \left[\bar{q}^{i+1}_{(s,a)} - \hat{Q}^i(s,a)\right]^2 \;=\; \alpha \left[\bar{q}^{i+1}_{(s,a)} - \hat{Q}^i(s,a)\right] \frac{\partial\, \hat{Q}^i(s,a)}{\partial w_{ji}}$$

in Richtung des für das Zustands-Aktionspaar (s, a) neu berechneten Q-Wertes

$$\bar{q}_{(s,a)}^{i+1} = r + \gamma \max_{a' \in A} \hat{Q}^i(s', a')$$

angepasst.

Diese auch als "direkte" [4] Methoden bezeichnete Art der approximativer Verfahren mit sofortigen online Aktualisierungen sind in der praktischen Anwendung aber ganz und gar nicht problemfrei. Zwar wurden früh einige beeindruckende praktische Erfolge erzielt [29, 91, 128, 129, 160, 161], allgemein gültige Konvergenzaussagen wie beim exakten DP oder Q-Lernen fehlen allerdings. Die Konvergenzeigenschaften sind vielmehr von der konkreten Modellierung und der Kombination von Lernregel und Approximator abhängig. Vielfach ist es selbst für einfache MDPs [20] schwer oder sogar unmöglich, Divergenz sicher zu verhindern [4, 20, 52, 54, 97]. Die wenigen erfolgreichen Verfahren sind in der Praxis meist sehr schwer einzustellen. Aus diesen Gründen und weil diese online Verfahren, gemessen an den notwendigen Interaktionen, in der Praxis sehr langsam lernen, hat sich mit den batch RL-Methoden in jüngster Zeit ein alternativer Ansatz durchgesetzt.

2.2.7 Batch Reinforcement Lernen

In aktuellen batch RL-Verfahren spielen zwei Ideen eine zentrale Rolle. Zunächst basieren die batch RL-Verfahren auf der auf Gordon [51] zurückgehenden Idee, die Vorgänge dynamisches Programmieren und Approximation der Wertfunktion voneinander zu trennen. Anders als bei den direkten online Verfahren, bei denen jede berechnete DP-Aktualisierung sofort im Approximator "abgespeichert" wird, wird in Gordons fitted Ansatz zunächst auf einer endlichen – und damit handhabbaren – Teilmenge $\mathcal{F} \subset S$ mit $\mathcal{F} = \{s_i | i = 1, \ldots, n\}$ von n festen Stützstellen $s_i \in S$ im unendlichen und potenziell mehrdimensionalen Zustandsraum $S \subseteq \mathbb{R}^m$ eine DP-Aktualisierung vorgenommen. Im Falle von Gordons Fitted Value Iteration (FVI) [51] bedeutet dies eine Berechnung der Aktualisierung nach

$$\begin{aligned} \bar{v}_i^{k+1} &\leftarrow \max_{a \in A} E\left[r + \gamma \hat{V}^k(s_{t+1}) \middle| s_t = s_i, a_t = a\right] \\ &= \max_{a \in A} \sum_{s' \in S} T(s_i, a, s') \left[R(s, a, s') + \gamma \hat{V}^k(s')\right] \end{aligned} \tag{2.17}$$

für alle Stützstellen $s_i \in \mathcal{F}$, wobei $\hat{V}^k(s)$ die Approximation der Wertfunktion $V^k(s)$ nach k Schritten dynamischen Programmierens und $\bar{v}_i^{k+1}$ die Schätzung des Zustandswertes $V^{k+1}(s_i)$ an der Stützstelle s_i ist.[1] Die an den Stützstellen berechneten Zielwerte bilden einen Satz von n Trainingsdaten

$$\mathcal{P} = \{(s_i; \bar{v}_i^{k+1}) | s_i \in \mathcal{F}\}$$

[1] Eigentlich müsste in (2.17) über die Übergangswahrscheinlichkeiten integriert werden. Die von Gordon verwendete Summenbildung macht nur Sinn, wenn von deterministischen Übrgängen, großen aber endlichen Zustandsmengen oder von einer zwar unendlichen Zustandsmenge, aber jeweils nur wenigen möglichen Folgezuständen mit $T(s_t, a_t, s_{t+1}) > 0$ ausgegangen wird.

mit Eingabevektoren s_i und zugehörigen Zielwerten $\bar{v}_i^{k+1}$ – für jede Stützstelle $s_i \in \mathcal{F}$ genau ein Beispiel $(s_i; \bar{v}_i^{k+1})$ – auf die im Anschluß ein Funktionsapproximator mittels überwachten Lernens trainiert wird (das sogenannte "fitting"). Der trainierte Funktionsapproximator realisiert die Funktion $\hat{V}^{k+1}(s)$ als Approximation der $k+1$-ten Wertfunktion $V^{k+1}(s)$. In diesem synchronen Verfahren wird somit immer zuerst je ein Schritt dynamischen Programmierens auf sämtlichen Stützstellen mit Hilfe der alten Schätzung $\hat{V}^k(s)$ der Wertfunktion berechnet, bevor die Schätzung der Wertfunktion durch das Training eines neuen Funktionsapproximators verändert wird.

Der synchrone "fitted" Ansatz erweist sich als wesentlich stabiler als vergleichbare online Verfahren [51, 54]. Theoretisch konnte von Gordon für den Einsatz eines Funktionsapproximationsverfahren A mit dem Abbildungsoperator M_A, der eine gegebene Zielfunktion f auf die gelernte Funktion $\hat{f}$ abbildet, die Konvergenz der Folge $(\hat{V}^i)$ der approximierten Wertfunktionen $\hat{V}^i$ bei Problemen mit diskontierter Belohnung gegen eine Funktion $\hat{V}$ nachgewiesen werden, sofern M_A eine Kontraktion (bzw. nicht-Expansion) in Maximum-Norm mit $|\hat{f}(x) - \hat{g}(x)| \leq |f(x) - g(x)|$ für alle $x \in \mathbb{R}^m$ ist. Der Abstand des Ergebnisses $\hat{V}$ von der optimalen Wertfunktion V^* ist dabei durch

$$||V^* - \hat{V}||_\infty \leq 2\epsilon + \frac{2\gamma\epsilon}{1-\gamma}$$

beschränkt, wobei $\epsilon = ||V^* - V^A||_\infty$ der Abstand des an V^* nächstgelegenen Fixpunktes V^A von M_A in Maixmum-Norm ist und quasi ein Maß dafür darstellt, wie gut der gewählte Funktionsapproximator die Struktur der optimalen Wertfunktion prinzipiell erfassen kann. Hat V^* große Sprünge (Diskontinuitäten), deren Lage vom eingesetzten Funktionsapproximator nicht gut erfasst werden kann, verliert diese Schranke aufgrund der Abhängigkeit vom maximalen Fehler aber an Aussagekraft.Diese theoretischen Ergebnisse werden im Zusammenhang mit der Analyse von DFQ in Kapitel 4 noch näher erläutert.

Große Aufmerksamkeit erhielt dieser Ansatz in jüngster Zeit durch die Kombination mit der Idee des "Experience Replays" [91], die zu eine Reihe neuer Verfahren geführt hat. In diesen neuen, speicherbasierten Verfahren werden Aktualisierungen nicht mehr modellbasiert an den vorgegebenen Stützstellen, sondern – der anderen zentralen Idee folgend – modellfrei entlang beobachteter Zustandsübergänge vorgenommen. In der Interaktion gewonnene Erfahrungen in Form von Zustandsübergängen (s, a, r, s') werden nicht nur für eine einzige Aktualisierung der Strategie bzw. Wertfunktion herangezogen, sondern für spätere, sich kontinuierlich wiederholende Verwendung gespeichert. Da die Aktualisierungen immer noch ausschließlich auf vom System gezogenen Übergängen vorgenommen werden, spiegeln sich die Übergangswahrscheinlichkeiten, nach denen die Beobachtungen gezogen werden, weiterhin im erzeugten Datensatz wider, so dass auch stochastisch approximative Verfahren wie das Q-Lernen problemlos mit dem Experience Replay kombiniert werden können. Zu den populärsten dieser neuen batch Vefahren sind Least Squares Policy Iteration (LSPI) [79], Fitted Q-Iteration (FQI) [39], und Neural Fitted Q (NFQ) [130] zu zählen. In [112, 113] wurden von Ormoneit et al. zentrale, in FQI umgesetzte Ideen beschrieben und wichtige theoretische Grundlagen für diese übergangsbasierten Verfahren gelegt. Mit Hilfe dieser jungen Verfahren

wurden bereits einige beeindruckende Erfolge an praktischen Problemstellungen erzielt [47, 74, 133, 134, 171]. Folgende Veranschaulichung konzentriert sich beispielhaft auf die Variante des FQI-Algorithmus für endliche Aktionsmengen, da dieses Verfahren bei der Enticklung von DFQ in Kapitel 3 eine ganz zentrale Rolle spielen wird.

2.2.8 Fitted Q-Iteration

Gegeben eine Menge $\mathcal{F} = \{(s_t, a_t, r_{t+1}, s_{t+1}) | t = 1, ..., p\}$ von p in der Interaktion mit einem System gezogenen Transitionen $(s_t, a_t, r_{t+1}, s_{t+1})$ und einem initialen Q-Wert $\bar{q}^0$, bei Ernst $\bar{q}^0 = 0$ [39], wird im FQI-Verfahren über folgende Schritte iteriert:

---

1. Initialisierung Erzeugung einer initialen Approximation $\hat{Q}^0(s, a)$ mit

$$\hat{Q}^0(s, a) = \bar{q}^0 \quad \forall (s, a) \in S \times A$$

und Setzen des Iterationszählers i auf Null $i \leftarrow 0$.

2. Dynamisches Programmieren Konstruktion einer Menge von Trainingsbeispielen $\mathcal{P}^{i+1} = \{(s_t, a_t; \bar{q}_t^{i+1}) | t = 1, ..., p\}$ durch Vollziehen eines Schritts des dynamischen Programmierens "entlang" aller in $\mathcal{F}$ enthaltenen p Transitionen. Dazu wird ein Zielwert $\bar{q}_t^{i+1}$ für jedes der p Zustandsaktionspaare (s_t, a_t) berechnet, die den Anfangspunkten und gewählten Aktionen der in $\mathcal{F}$ enthaltenen Zustandsübergängen entsprechen. Die Berechnung des Zielwertes geschieht durch Anwendung der Aktualisierungsvorschrift aus dem Q-Lernen mit

$$\bar{q}_t^{i+1} \leftarrow r_{t+1} + \gamma \max_{a' \in A} \hat{Q}^i(s_{t+1}, a') \quad \forall (s_t, a_t, r_{t+1}, s_{t+1}) \in \mathcal{F}\,, \tag{2.18}$$

wobei $\hat{Q}^i : S \times A \mapsto \mathbb{R}$ die aktuelle Approximation der Q-Funktion ist. Für direkte Übergänge in einen absorbierenden Terminalzustand (kürzester Pfad-Probleme) muss statt (2.18) die Aktualisierung $\bar{q}_t^{i+1} \leftarrow r_{t+1}$ verwendet werden(siehe auch [53]), da per Definition (siehe Abschnitt 2.2.2) im absorbierenden Terminalzustand s_t keine weiteren Kosten anfallen.

3. Überwachtes Lernen Training eines Funktionsapproximators auf den Trainingsdaten $\mathcal{P}^{i+1}$ mittels überwachten Lernens. Die vom Funktionsapproximator realisierte Funktion $\hat{Q}^{i+1}$ ist dann die Approximation der Q-Funktion Q^{i+1} nach $i+1$ Schritten dynamischen Programmierens.

4. Schleife Erhöhung des Schleifenzählers $i \leftarrow i + 1$ und Fortfahren mit Schritt 2, solange das Abbruchkriterium nicht erreicht ist. Als Abbruchkriterien kommen das Erreichen einer vorgebenen Anzahl von Iterationen oder das Unterschreiten einer oberen Schranke für den Bellman-Restwert $||\hat{Q}^{i+1} - \hat{Q}^i||_\infty$ in Frage.

Für $0 \leq \gamma < 1$ lässt sich zeigen, dass die Folge der Q-Funktionen $(\hat{Q}^i)$ in der Maximum–Norm durch

$$||\hat{Q}^i(s,a)||_\infty \leq \frac{B_r}{1-\gamma} \quad \forall i \in N \tag{2.19}$$

beschränkt ist, sofern mit $q^0 = 0$ gestartet wird [39]. B_r ist dabei eine obere Schranke für die maximale sofortige Belohnung r. Auf Basis der Ergebnisse von Ormoneit und Sen [113] lässt sich zudem – ebenfalls für $\gamma < 1$ – zeigen, dass der FQI-Algorithmus von jedem beliebigen Ausgangspunkt $\hat{Q}^0$ gegen eine Funktion $\hat{Q}$ als eindeutigen Fixpunkt konvergiert [39, 113]. Die Nachweise beider Aussagen bauen allerdings darauf, dass die Wertfunktion je Aktion auf Basis aller Trainingsbeispiele $\mathcal{P}_a = \{(s^j, a^j, \bar{q}^j) \in \mathcal{P} |\ a^j = a\}$ in Schritt 3 durch einen kernelbasierten Approximator der Form

$$f_a(s) = \sum_{j=1}^{|\mathcal{P}_a|} k(s^j, s)\bar{q}^j \tag{2.20}$$

geschätzt wird, wobei für den gewichtenden Kernel $k(s^l, s)$ die Normalisierungsbedingung

$$\sum_{j=1}^{|\mathcal{P}_a|} k(s^j, s) = 1 \quad \forall s \in S$$

gilt. Der erlernte Q-Wert $\hat{Q}(s,a)$ eines Zustandsaktionspaars kann aus den in separaten Funktionsapproximatoren f_a für alle $a \in A$ gespeicherten Aktionswertfunktionen leicht berechnet werden, indem der Funktionswert des a entsprechenden Approximators f_a zurückgegeben wird, also $\hat{Q}(s,a) = f_a(s)$ [39]. Unter diese Formulierung des kernelbasierten Funktionsapproximators fallen auch Gordons Averager. Andere beliebte Approximatoren wie lineare Regression [53], RBFs [97], Locally Weighted Regression und MLPs [20] scheiden aber aus.[1]

Diese und weitere theoretischen Befunde werden in Kapitel 4 eingehender behandelt, das sich ausführlich mit den theoretischen Ergebnissen zum batch RL und den Arbeiten von Gordon und Ormoneit et al. befasst. Dort wird auch die hier verwendete Methode näher erläurtert, die Q-Funktion durch einen separaten Funktionsapproximator je Aktion zu repräsentieren.

2.2.9 Batch Reinforcement Lernen mit Exploration

Ursprünglich wurde FQI von Ernst als reines offline Verfahren konzipiert, das auf einer vorgegebenen, festen Menge von Übergängen arbeitet [39]. Eine Möglichkeit, dieses Verfahren dennoch mit einer Exploration zu verbinden – zum Beispiel um das Ergebnis im

[1] Auch die beliebten und lange für stabil gehaltenen Multigrids (CMACs) werden schon durch Gordons Aussagen nicht erfasst [53] und erweisen sich als allgemein instabil, was von Merke auch formal durch den Nachweis der Verletzung eines notwendigen Kriteriums für die Stabilität einer eng verwandten, synchronen Aktualisierungsregel belegt wurde [97].

Sinne von [113] weiter zu verbessern – ist in Abbildung 2.7 grafisch dargestellt. Aus der in der inneren Schleife mittels FQI bestimmten Q-Funktion wird eine Explorationsstrategie abgeleitet. Das kann zum Beispiel durch Auswertung nach (2.15) und Anwendung der ϵ-greedy Exploration [156] in der äußeren Schleife geschehen. Alle in der äußeren Schleife in der Interaktion mit dem System gesammelten Transitionen werden zum bestehenden Datensatz $\mathcal{F}$ hinzugefügt. Nach einiger Zeit wird dann auf dem so um neue Beobachtungen erweiterten Datensatz eine Wertfunktion von Grund auf neu berechnet.

Batch RL-Verfahren dieses Typs werden in der Folge als "semi-online" Verfahren bezeichnet, da sie sich von außen betrachtet wie online Verfahren verhalten, das System selbständig explorieren und währenddessen ihre Strategie verbessern. Intern speichern sie aber im Gegensatz zu den online Verfahren alle beobachteten Übergänge ab und führen Trainingsverfahren aus dem batch RL durch. Für diese spezielle Variante des batch RLs sehr interessante theoretische Ergebnisse sind bei Ormoneit und Sen [113] zu finden und werden unter dem Stichwort "Konsistenz" im Kapitel 4 ausführlich diskutiert.

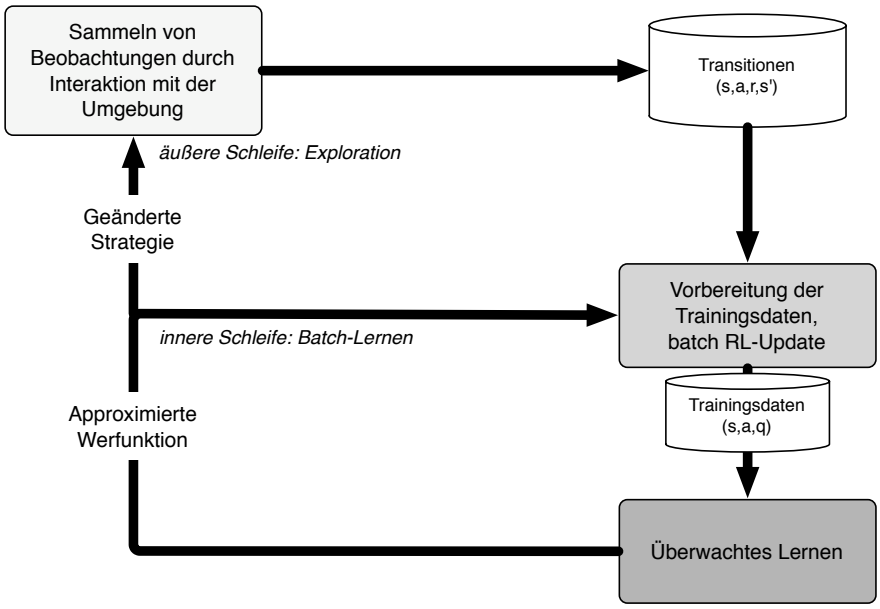


Abbildung 2.7: Batch RL mit Exploration nach Riedmiller et al. [134]. In der inneren Schleife wird das approximative dynamische Programmieren vollzogen. Von der berechneten Q-Funktion wird anschließend eine Explorationsstrategie abgeleitet, mit der in der äußeren Schleife neue Transitionen gesammelt werden. Nach einiger Zeit wird die innere Schleife mit dem so erweiterten Datensatz aufgerufen.

3

DFQ: Optimierendes Lernen auf hochdimensionalen Observationen

In diesem Kapitel wird der neue Deep Fitted Q-Algorithmus zum Lernen in visuomotorischen Problemstellungen vorgestellt. Insbesondere wird eine Übersicht über den generellen Ablauf gegeben und dann eingehend beschrieben, mit welchen Techniken sich die automatische Generierung von Merkmalsräumen so in den batch RL-Ablauf einbetten lässt, dass einerseits die Stabilität des Lernvorganges möglichst wenig beeinträchtigt wird und gleichzeitig die hohe Dateneffizienz der batch RL-Verfahren bestehen bleibt. Die Konvergenz des Algorithmus wird in einem eigenen Kapitel (Kapitel 4) untersucht, die konkreten Realisierungen der zwei zentralen Module des Algorithmus – das Erlernen geeigneter Merkmalsräume und die Approximation der Wertfunktion – werden in den Kapiteln 5 und 6 eingehend erläutert.

3.1 Motivation

Batch Reinforcement Lernverfahren haben einige Erfolge an realen Systemen ermöglicht. Für den Einsatz direkt auf realen Bilddaten in visuellen Lernproblemen sind sie aber aus Komplexitätsgründen dennoch nicht geeignet. Hier wird nun die Idee verfolgt – wie in der Einleitung beschriebenen, klassischen Lösungsansatz für das visuomotorische Lernen – zunächst einen Merkmalsraum zu konstruieren, in den die Bilder möglichst informationserhaltend abgebildet werden. Nicht auf den Bildern, aber auf den zugehörigen Merkmalsvektoren können dann Verfahren wie Ernsts FQI [39] erfolgreich eingesetzt werden. Der entscheidende Schritt, der in diesem Kapitel vollzogen wird, ist es nun, die Konstruktion des Merkmalsraums selbst ebenfalls voll zu automatisieren und vollständig in das Lernverfahren zu integrieren. Dies geschieht über die Einführung einer dem batch Reinforcement Lernverfahren vorgeschalteten Stufe zur automatischen Konstruktion eines Merkmalsraums auf Basis der zur Verfügung stehenden Beobachtungen.

Beim Entwurf des Ablaufrahmens für eine Verknüpfung von Reinforcement Lernen und automatischer Konstruktion von Merkmalsräumen ist eine wichtige Frage, wie genau diese beiden Verfahren in einem Gesamtablauf integriert und aufeinander abge-

stimmt werden können, so dass sowohl die Stabilität als auch die Effizienz gewährleistet sind. Welche Daten müssen gesammelt werden, wann können die Aktualisierungen des Merkmalsraums einerseits und der Strategie andererseits am Besten vorgenommen werden?

In den folgenden Abschnitten wird der unter diesen Gesichtspunkten entwickelte batch RL-Algorithmus "DFQ" für "deep fitted q-iteration" beschrieben, in dem tiefe Autoencodernetze zum Erlernen von Merkmalsräumen mit Ernsts Fitted Q-Iterations zum Erlernen von Strategien verknüpft werden. Im Abschnitt 3.2 wird zunächst die klassischen Agenten-Umgebungs-Schleife [142, 156] für die hier betrachteten visuomotorischen Problemstellungen erweitert. An Hand dieser Erweiterung wird diskutiert, wie sich die einzelnen Verfahren in den Ablauf bei der Interaktion mit dem System eingliedern. Während sich diese Beschreibung quasi auf die Anwendungsphase des fertig trainierten Agenten beschränkt, wird im darauf folgenden Abschnitt 3.3 schließlich der eigentliche Lernvorgang beschrieben.

Im Zusammenhang mit einer semi-online Variante des DFQ Algorithmus mit kontinuierlicher Optimierung des Merkmalsraumes tritt die Frage auf, ob eine erlernte Strategie zu einer veränderten Einbettung mit anderen Merkmalen übertragen werden kann oder ob das erworbene Wissen nach jeder Adaptation des Merkmalsraums verworfen und von vorne begonnen werden muss. Die in Antwort auf diese Frage entwickelten Techniken zur Steigerung der Dateneffizienz werden zusammen mit Optimierungen zur Verminderung des Rechenaufwands in Abschnitt 3.5 beschrieben. Die erzielte effiziente Kombination des tiefen Lernens mit batch RL-Verfahren stellt einen wichtigen Beitrag dieser Arbeit dar. Die hierbei ausdrücklich für visuomotorischen Problemstellungen erarbeiteten Ergebnisse sind zudem auch für ein breiteres Spektrum von Lernaufgaben interessant; sie können immer dann zum Einsatz kommen, wenn RL auf einem Zustandsraum eingesetzt werden soll, der während des Lernens nicht konstant bleibt, sondern adaptiert wird und seine Semantik ändert.

3.2 Struktur der Interaktion mit der Umgebung

Im bereits beschriebenen allgemeinen Reinforcement Lernproblem nach Sutton und Barto [156] interagiert ein Agent mit der Umgebung in diskreten Zeitschritten t, zu denen er einen Zustand s_t und eine Belohnung r_t wahrnimmt, um dann mit einer Aktion a_t zu antworten. Im hier betrachteten visuellen Reinforcement Lernproblem ist der Zustand s_t des Systems nun aber nicht direkt durch den Agenten beobachtbar. Stattdessen erhält der Agent in jedem Zeitschritt t eine hochdimensionale, kontinuierliche Observation $o_t \in \mathcal{O}$. Hier handelt es sich bei den Observationen um in reellwertigen Vektoren kodierte Grauwertbilder $o \in [0,1]^n$, wobei n der Dimension der Bildmatrix als Produkt der Anzahl ihrer Zeilen und Spalten entspricht.

Typischerweise übersteigt die Dimension n der Bilder um ein Vielfaches die Komplexität, die noch mittels aktueller batch RL-Verfahren in der Praxis handhabbar wäre. Aus diesem Grund wird das RL-Verfahren in dem hier vorgeschlagenen Lösungsansatz nicht direkt auf die Observationen o_t, sondern auf einen Merkmalsvektor $z_t \in \mathbb{R}^m$ an-

gewendet, der im Idealfall die gleiche Information wie o_t in wesentlich kompakterer Form mit $m \ll n$ repräsentiert. Berechnet wird dieser Merkmalsvektor z_t direkt aus der Observation o_t. Auch wenn hier ganz konkret auf die Berechnung der Merkmalsvektoren mit Hilfe eines tiefen Encoders nach $z_t = \mathrm{ENC}(o_t; W)$ abgezielt wird, sind aus Sicht des Rahmenablaufs prinzipiell viele verschiedene Berechnungsmethoden denk- und einsetzbar. In Abbildung 3.1 ist der in dieser Weise für das visuelle Reinforcement Lernproblem veränderte Ablauf aus Abbildung 2.7 dargestellt, wobei auch die Module des hier propagierten Ablaufrahmens an passender Stelle eingeordnet wurden.

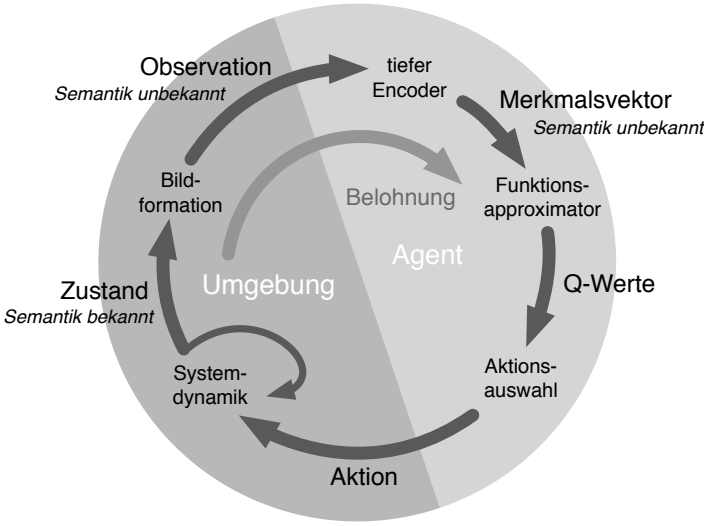


Abbildung 3.1: Erweiterte Interaktionsschleife zwischen Agent und Umgebung mit Merkmalsextraktion.

Im Rahmen dieser Arbeit wird durchweg davon ausgegangen, dass auf der Seite der Umgebung eine – auch rein hypothetische – Beschreibung des Systemzustands s_t existiert, die den relevanten Systemzustand vollständig beschreibt und insbesondere der Markov-Eigenschaft (2.2) genügt. Wie im gewöhnlichen MDP ergibt sich auch im erweiterten Prozess der Folgezustand s_{t+1} des Systems allein aus dem aktuellen Zustand s_t und der durch den Agenten gewählten Aktion a_t. Dieser der Umgebung zugeordnete Zustandsübergangsprozess ist vom Agenten aber nicht einsehbar. Auch die Interna des Bildformationsprozesses, der aus dem Systemzustand s_t eine Observation o_t erzeugt, sind dem Agenten unbekannt und können auch nicht vom ihm beeinflusst werden. Darunter fallen sowohl das Aussehen des Systems und die Umgebungsbedingungen zum Aufnahmezeitpunkt als auch der eigentliche Bildaufnahmevorgang mit der Kamera. Deshalb wird die Bildformation in dieser Modellierung des visuomotorischen Lernproblems ebenfalls vollständig der Umgebung zugeordnet (dunkelgraue, linke Hälfte der

Abbildung 3.1). Der Bildformationsprozess selbst kann dabei genau wie die Systemübergänge auch stochastischer Natur sein. Die Wahrscheinlichkeit $P(o_t = o|s_t = s)$ eine bestimmte Observation o_t zum Zeitpunkt t zu erhalten, unterliegt im weiteren Verlauf aber unter anderem der milden Einschränkung, nur vom aktuellen Zustand s_t abhängig zu sein (siehe Beschränkung 1 weiter unten).

Der vollen Kontrolle des Agenten (hellgraue, rechte Hälfte der Abbildung 3.1) unterliegt in dem hier verfolgten Ansatz neben der Approximation der Wertfunktion auch der Vorgang der Merkmalsextraktion, also die Berechnung von z_t aus der Observation o_t. Auf Basis des aktuellen Merkmalsvektors trifft der Agent dann mit Hilfe der approximierten Wertfunktion die Auswahl einer geeigneten Aktion. Da in diesem Modell das System aus Sicht des Agenten vollständig opak und beim Agenten im Allgemeinen kein Zustandsübergangsmodell vorhanden ist, ist der Einsatz eines modellfreien Ansatzes erforderlich und es werden, wie dargestellt, optimale Q-Werte approximiert.

Das beschriebene Vorgehen kennzeichnet zwei Beschränkungen:

1. Linkseindeutige, informationserhaltende Bildformation: Die Forderung nach der Existenz einer Zustandsbeschreibung des Systems mit Markov-Eigenschaft führt dazu, dass das Lernproblem unter Kenntnis dieser Zustandsbeschreibung als MDP (S, A, T, R) ("Systemzustands-MDP") modellierbar wäre. Ob der auf diesem Systemzustand basierende, visuelle Entscheidungsprozess auch in dieselbe Klasse fällt oder ein schwerer zu lösender partiell observierbarer markovscher Entscheidungsprozess (POMDP) [100] ist, hängt allerdings vom Informationsgehalt der Observationen ab. Um das untersuchte Problem des Lernens auf hochdimensionalen Bilddaten nicht noch weiter zu komplizieren, wird in den folgenden Ausführungen ausschließlich davon ausgegangen, dass ein und derselbe Systemzustand zwar – zum Beispiel aufgrund von Bildrauschen – unterschiedliche Observationen hervorrufen kann, dass aber keine zwei verschiedenen Systemzustände dieselbe Observation hervorrufen können – also von einer Observation prinzipiell eindeutig auf den internen Systemzustand zurückgeschlossen werden kann. Bei einer solchen Bildformation, die Systemzustände und Observationen in eine linkstotale, linkseindeutige Relation $\mathcal{R} \subseteq S \times O$ setzt, bleiben alle relevanten Informationen erhalten und die Systemübergänge auf Basis der Observationen sind ebenfalls markov. In diesem Fall lässt sich ein zweites, zum Zustands-MDP korrespondierendes MDP (O, A, T_O, R_O) ("Observationen-MDP") auf Basis der Observationen aufstellen. Für die Übergangsfunktion T_O (und in einer analogen Formulierung für den Erwartungswert R_O) gilt

$$T_O(o, a, o') = \sum_{(s,s') \in S \times S} T(s, a, s') P(s_t = s|o_t = o) P(s_{t+1} = s'|o_{t+1} = o') \ ,$$

wobei in diesem Fall aufgrund der Linkseindeutigkeit der Bildformation $P(s_t = s|o_t = o) = 1$ für genau denjenigen Zustand $s \in S$, der als einziger in der Lage ist, die Observation o hervorzurufen, also $(s, o) \in \mathcal{R}$ und $P(s_t = s|o_t = o) = 0$ für alle anderen Zustände (gleiches gilt für den Zusammenhang s' und o').

2. **Externes Belohnungssignal:** Das Belohnungssignal muss notwendigerweise in dieser Modellierung ganz klassisch der Umgebung zugerechnet werden. Die Definition einer Belohnungsstruktur innerhalb des Agenten müsste nämlich in Ermangelung der Kenntnis des Systemzustandes auf den Observationen geschehen. Da die Semantik der Bilder aber vorab nicht bekannt und die Analyse der Bildinformationen Teil des Lernproblems selbst ist, scheidet diese Möglichkeit aus, sofern der Agent nicht über weitere Sensoren verfügt oder aber die Bilder von vornherein teilweise auswerten kann, um zumindest den Zielzustand oder die aktuellen Kosten detektieren zu können.

Im praktischen Einsatz hat die Einschränkung 2 auf externe Belohnungssignale vielfach keine größeren Auswirkungen. Entweder liefert das System bereits ein eindeutiges Feedback über einen anderen Kanal als die Kamera oder es wird eben doch eine spezielle Lösung auf Seiten des Agenten bereitgestellt, die nichts anderes leistet, als in Abwandlung des Kreislaufs dort im Agenten das Belohnungssignal an Hand der Bilder zu bestimmen.

Die Beschränkung in 1 führt dazu, dass wertfunktionsbasierte batch RL-Ansätze auf die Beobachtungen, bzw. auf die daraus gewonnenen Merkmalsvektoren angewendet werden können. Zwar mag die Lösung des Observationen-MDPs aufgrund der potenziell vielen möglichen Beobachtungen für jeden möglichen Systemzustand aufwändiger sein als die Lösung des ursprünglichen Systemzustands-MDPs, eine bezüglich des Observationen-MDPs optimale Strategie verhält sich unter Beschränkung 1 aber auch bezüglich des Systemzustands-MDPs optimal. Die Lösung von allgemeineren POMDPs ist Gegenstand eines eigenen Forschungszweiges und wird sinnvollerweise getrennt von dem Problem der automatischen Extraktion niedrigdimensionaler Zustandsbeschreibungen aus hochdimensionalen Beobachtungen betrachtet.

Nicht unerwähnt bleiben darf an dieser Stelle, dass durch die Forderung nach einer linkseindeutigen Bildformation mit markovschen Observationen zunächst (fast) alle dynamischen Systeme ausgeschlossen werden. So kann die Bewegung eines Objekts nicht allein von einem einzelnen Bild als statisches Abbild eines Augenblicks abgeleitet werden. Nach der prinzipiellen Darstellung und Erprobung der für das visuelle Lernen entwickelten Methoden wird in Kapitel 7 aber in einer praktischen Anwendung vorgeführt, wie sich auch dynamische Probleme durch eine einfache Erweiterung des gesteckten Rahmens lösen lassen, zum Beispiel über die Formulierung als MDP höherer Ordnung und die Betrachtung von Bildsequenzen.

Nachdem festgelegt ist, wie der Agent mit dem System interagiert, kann nun diskutiert werden, wie ein Agent in solchen Problemen zu lernen vermag.

3.3 Training des Agenten

Unterschieden werden der Ablauf als reines batch Trainingsverfahren und der Ablauf unter Einbindung der Exploration in den Lernvorgang.

3.3.1 Rahmenablauf ohne Exploration

Im reinen batch RL-Framework steht dem Lernverfahren gleich zu Beginn eine Menge von Übergangsdaten zur Verfügung, die auf eine unbekannte Weise vom System gewonnen wurden. Basierend auf diesem Datensatz soll die bestmögliche Strategie erlernt werden, die in der dann folgenden Anwendungsphase nicht weiter adaptiert wird. Es besteht also eine strenge Untergliederung zwischen Lernphase und Interaktionsphase; insbesondere ist es dem Agenten (bzw. dem Lernalgorithmus) nicht erlaubt, mit dem System zu interagieren und die Exploration selbständig zu steuern. Dies ist der Ablauf, wie er auch von Ernst in [39] für die Fitted Q-Iteration untersucht wurde.

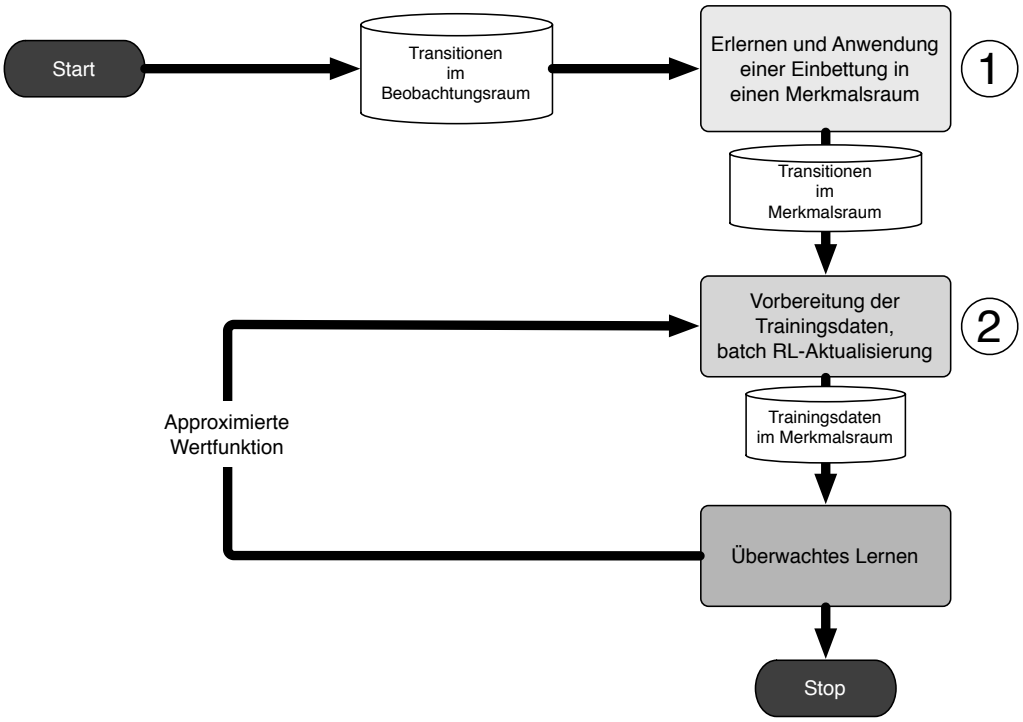


Abbildung 3.2: Grafische Darstellung des Ablaufrahmens für batch Reinforcement Lernen mit automatischer Merkmalsextraktion in der reinen offline Version ohne Exploration.

In einer solchen Problemstellung lassen sich die beiden Stufen Merkmalsextraktion und Strategielernen in offensichtlicher Weise, wie in Abbildung 3.2 dargestellt, rein sequentiell koppeln. Zunächst wird in Stufe 1 auf den Beobachtungen o in den Übergangsdaten $\mathcal{F}_O = \{(o_t, a_t, r_{t+1}, o_{t+1}) | t = 1, \ldots, p\}$ ein Merkmalsextraktor – zum Beispiel ein tiefer Autoencoder – trainiert, der einen Encoder $z = \text{ENC}(o)$ zur Berechnung der Merkmalsvektoren $z \in Z$, mit $Z = \mathbb{R}^n$, aus den Observationen $o \in O$ realisiert. Mit Hilfe dieses Encoders wird aus der Menge $\mathcal{F}_O$ eine Menge $\mathcal{F}_Z = \{(z_t, a_t, r_{t+1}, z_{t+1}) | t = 1, \ldots, p\}$ von Übergangsdaten im Merkmalsraum Z mit $z_t = \text{ENC}(o_t)$ berechnet. Anschließend wird auf Basis der so extrahierten und sich nicht mehr ändernden Merkmalsvektoren

in Stufe 2 mittels FQI oder anderen batch RL-Verfahren eine Wertfunktion auf $\mathcal{F}_Z$ approximiert, von der eine Strategie abgeleitet werden kann.

3.3.2 Rahmenablauf mit Exploration

Wie schon in Abschnitt 2.2.9 diskutiert wurde, ist es in der Praxis bei vielen Systemen schwierig, einen solchen Datensatz $\mathcal{F}_O$ allein mit der Zufallsstrategie zu sammeln und dennoch den Raum möglicher Beobachtungen ausreichend gut abzudecken. Oft lässt sich auf solchermaßen gesammelten Übergangsdaten keine gute, geschweige denn eine optimale Strategie erzielen. Auch die Encoder liefern für ungleichmäßige, lückenhafte Verteilungen der Observationen keine optimalen Ergebnisse. Um aber die initiale Startregion zu verlassen und ausreichend gut zu explorieren, ist oft bereits eine gute Strategie oder einiges an Vorwissen über das betrachtete System nötig. Daher wird in der Praxis häufig auch bei Verwendung von batch RL-Verfahren exploriert. Wie lassen sich nun die Encoder einbetten in die in Abbildung 2.7 dargestellten wechselnden Phasen von offline Aktualisierungen auf den abgespeicherten Übergängen und Phasen der Exploration unter Anwendung der erlernten Strategie (siehe Abschnitt 2.2.9)?

In Abbildung 3.3 ist die Einbettung des Trainings eines Merkmalextraktors in den batch RL-Ablauf mit wiederkehrender Exploration aus Abschnitt 2.2.9 dargestellt. In der inneren Schleife dieser semi-online Variante findet das gewöhnliche, bereits beschriebene batch Reinforcement Lernen statt, zum Beispiel mit Hilfe des FQI-Algorithmus. Nur wird das dynamische Programmieren in Stufe 2 nicht wie im klassichen Fall auf den Systemzuständen x bzw. den hochdimensionalen Beobachtungen o, sondern auf den niedrigdimensionalen Merkmalsvektoren $z = \mathrm{ENC}^i(o)$ durchgeführt.

In der äußeren Schleife verwendet der Agent die aktuelle Approxmiation der Q-Funktion, um weitere Übergänge $(o_t, a_t, r_{t+1}, o_{t+1})$ vom System zu sammeln, zum Beispiel, indem er von der Wertfunktion eine Explorationsstrategie per ϵ-greedy Auswertung [156] ableitet. Die in Episoden der Interaktion gesammelten und zwischengespeicherten Daten werden verwendet, um in 1a einen gänzlich neuen Merkmalsextraktor ENC^{i+1} zu trainieren bzw. den bestehenden mit Hilfe neu gewonnener Daten zu verbessern. In Schritt 1b wird der aktuelle Encoder verwendet, um die Übergänge in den Merkmalsraum zu überführen und als Menge der beobachteten Übergänge im Merkmalsraum $\mathcal{F}_Z$ an die innere Schleife weiterzugeben.

3.3.3 Schnittstellen und Rückkopplung zwischen den beiden Stufen

Zwischen den beiden Stufen in der inneren und in der äußeren Schleife bestehen klar definierte Schnittstellen, die in der Praxis eine saubere Modularisierung ermöglichen:

- Die äußere Schleife liefert ausschließlich eine Menge von in einen Merkmalsraum übersetzten Transitionen $\mathcal{F}_Z$ an die innere Schleife. Wie die Übergänge gesammelt und wie die Merkmalsvektoren aus den Observationen berechnet werden, spielt dabei für den batch RL-Algorithmus in der inneren Schleife keine Rolle. Insbesondere muss auch der Encoder in der inneren Schleife zur Durchführung der DP-Schritte nicht bekannt sein (siehe Abschnitt 3.4).

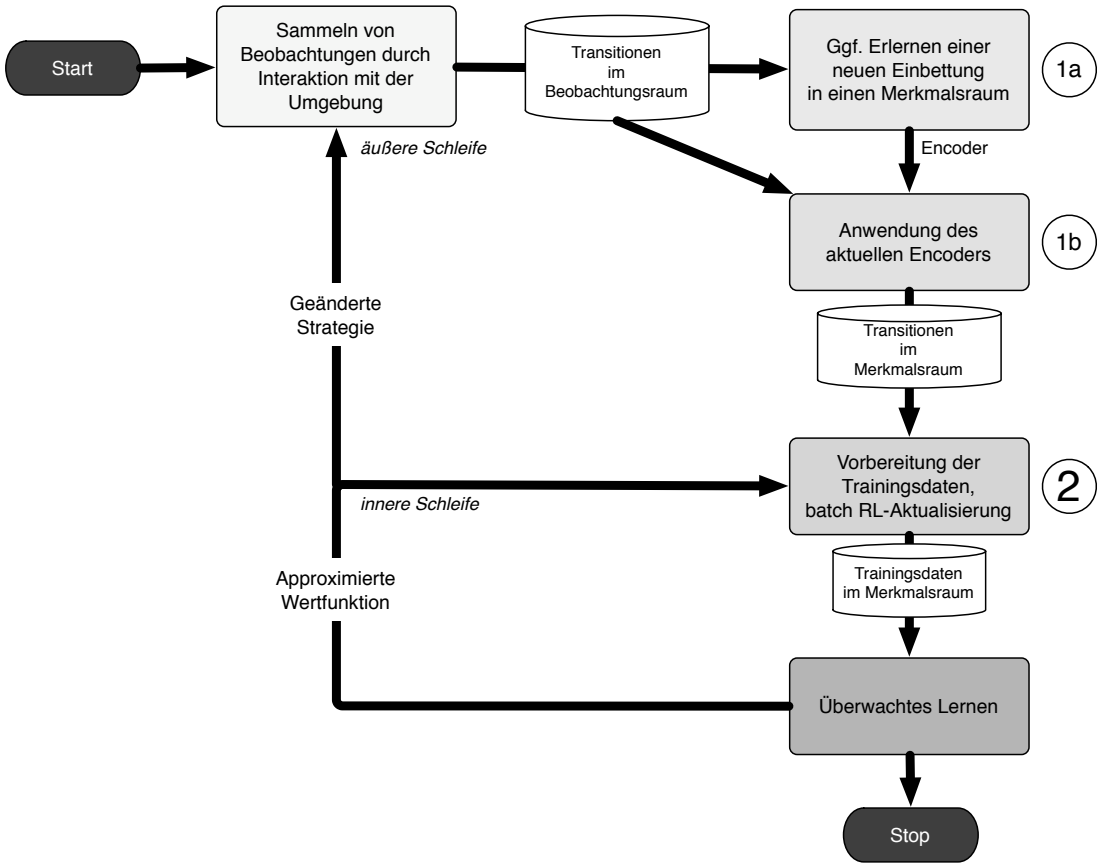


Abbildung 3.3: Grafische Darstellung des Ablaufrahmens für batch Reinforcement Lernen mit Merkmalsextraktion und episodischer Exploration.

- Die innere Schleife erzeugt auf Basis der Übergänge im Merkmalsraum eine Approximation der optimalen Wertfunktion und liefert diese an die äußere Schleife zurück. Dort kann diese Approximation verwendet werden, um eine Strategie zur Steuerung und Exploration des Systems abzuleiten. Mit Hilfe des dort verfügbaren Encoders können neue Beobachtungen in den Merkmalsraum übertragen und mit Hilfe der approximierten Wertfunktion, bzw. der abgeleiteten Strategie $\pi : Z \mapsto A$ ausgewertet werden.

Bei dieser klaren Kapselung kommt eine nützliche Eigenschaft der batch Verfahren zum Tragen: Da die batch RL-Verfahren nicht selbst mit dem System interagieren müssen, sondern auf einer festen Menge von Übergängen arbeiten, benötigen sie keine Kenntnis über die Gewinnung der Observationen und die Extraktion der Merkmalsvektoren. Erlernen der Strategie und Konstruktion der Merkmalsräume können demnach als weitestgehend eigenständige Probleme behandelt werden.

Eine gegenseitige Beeinflussung der beiden Verfahren ergibt sich aber dennoch schon allein aus der gemeinsame Nutzung der beobachteten Transitionen. Auch der Austausch von anderen Informationen an den Schnittstellen führt insbesondere bei der semi-online Variante zu einer gewissen (Rück-)Kopplung zwischen RL-Verfahren und Methode zur Merkmalsgewinnung:

- Einerseits beeinflusst die von der in Stufe 2 erlernten Wertfunktion abgeleitete Explorationsstrategie, welche Beobachtungen in der nächsten Explorationsphase eingesammelt werden und damit beim Erlernen eines Merkmalsraums zur Verfügung stehen. Bei Problemstellungen wie dem Mountain-Car oder dem inversen Pendel (Aufgabe: Aufschwingen) wird bereits eine recht gute Strategie benötigt, um die Startregion überhaupt verlassen und auch Beobachtungen in der Nähe des Zielzustandes sammeln zu können.

- Andererseits hängt von der Qualtiät des Autoencoders und den von ihm erzeugten Merkmalsvektoren ganz entscheidend ab, in welchen Bereichen des Zustandsraums überhaupt eine Wertfunktion approximiert werden kann. In schlecht explorierten Bereichen, wo der Autoencoder für gewöhnlich keine guten Merkmalsvektoren liefert, können die einzelnen Zustände unter Umständen nicht an Hand der erzeugten Merkmalsvektoren voneinander unterschieden werden. Daher kann in diesen Bereichen auch durch einen noch so guten RL-Ansatz keine sinnvolle Strategie erlernt werden.

Konstruktion des Merkmalsraums und Verbessern der Wertfunktion müssen in der semi-online Variante also Hand in Hand gehen. Mit zunehmender Ausbreitung der Abdeckung des Observationsraums mit Beobachtungen muss sich auch der gut in den Merkmalsraum abgebildete Teilbereich des Observationsraums vergrößern. Nur in diesem Bereich mit "stabilen" Merkmalsvektoren kann mittels DP-Techniken eine Wertfunktion approximiert und eine Strategie abgeleitet werden, die dann wiederum beim Ausdehnen der "Randbereiche" des explorierten und kodierbaren Beobachtungsraums hilft. Wie gut dieser Wechsel zwischen Optimierung des Merkmalsraums und Verbesserung der Strategie in der Praxis funktioniert, wird im Kapitel 7 an einem Labyrinthbeispiel empirisch untersucht, in dem ein schmaler Flaschenhals zu einem Zielraum sicher erkannt und durchschritten werden muss.

3.4 Deep Fitted Q-Iteration (DFQ)

Nach dieser Beschreibung des allgemeinen Ablaufrahmens für die Kombination von automatischer Merkmalsextraktion und batch RL-Verfahren wird nun beschrieben, wie in einem konkreten Algorithmus die Merkmalsextraktion in Stufe 1 mittels tiefer Autoencoder und das Erlernen der Wertfunktion in Stufe 2 mit modellfreier Fitted Q-Iteration realisiert werden können. Gegeben einen initialen Q-Wert $\bar{q}^0$ und eine maximale Episodenlänge arbeitet der Agorithmus "Deep Fitted Q-Iteration" (DFQ) die folgenden Schritte ab:

A. Initialisierung Setzen eines Schleifenzählers auf Null: $k \leftarrow 0$. Setzen eines Transitionszählers auf Null: $p \leftarrow 0$. Erzeugung einer initialen Explorationsstrategie $\pi^0 : Z \mapsto A$ und eines initialen Encoders ENC : $O \mapsto_{W^0} Z$. Beginn mit einer leeren Menge von Übergängen $\mathcal{F}_O = \emptyset$

B. Exploration Episodische Interaktion mit dem System über eine vorgegebene maximale Episodenlänge oder bis ein Terminalzustand erreicht ist. Durchführung folgender Schritte in jedem Zeitschritt t für jede von der Umgebung kommende Beobachtung o_t eines nicht-Terminalzustands:

1. Berechnung des Merkmalsvektors $z_t = \text{ENC}(o_t; W^k)$
2. Auswahl und Anwendung der Aktion nach $a_t \leftarrow \pi^k(z_t)$
3. Sobald Rückmeldung von Umgebung, Abspeichern des vollständigen Überganges $\mathcal{F}_O \leftarrow \mathcal{F}_O \cup (o_t, a_t, r_{t+1}, o_{t+1})$ und Erhöhung des Transitionszählers $p \leftarrow p + 1$ mit jeder gespeicherten Transition

C. Encoder lernen Training der Encodergewichte W^{k+1} in einem neuen tiefen Autoencoder DEC(ENC($\cdot$; W^{k+1}); W) auf den p in $\mathcal{F}_O$ enthaltenen Observationen (Details in Kapitel 5). Ableitung des neuen Encoders ENC($\cdot$; W^{k+1}) und Erhöhung des Episodenzählers $k \leftarrow k + 1$

D. Merkmalsvektoren berechnen Anwendung des Encoders ENC($\cdot$; W^k) auf alle p Viertupel $(o_t, a_t, r_{t+1}.o_{t+1}) \in \mathcal{F}_O$, um sie in den Merkmalsraum Z zu überführen und so die Menge $\mathcal{F}_Z = \{(z_t, a_t, r_{t+1}, z_{t+1}) |\ t = 1, \ldots, p\}$ mit $z_t = \text{ENC}(o_t; W^k)$ zu erhalten.

E. Innere Schleife: FQI-Algorithmus Aufruf des FQI-Algorithmus (vgl. Abschnitt 2.2.8) mit der Menge $\mathcal{F}_Z$ der Übergänge im Merkmalsraum und dem initialen Q-Wert $\bar{q}^0$:

1. Initialisierung Erzeugung einer initialen Approximation $\hat{Q}^0(z, a)$ mit

$$\hat{Q}^0(z, a) = \bar{q}^0 \quad \forall (z, a) \in Z \times A$$

und Setzen des Iterationszählers i auf Null: $i \leftarrow 0$.

2. Dynamisches Programmieren Konstruktion der Trainingsmenge $\mathcal{P}^{i+1} = \{(z_t, a_t; \bar{q}_t^{i+1}) | t = 1, ..., p\}$ mit $\bar{q}_t^{i+1} = r_{t+1} + \gamma \max_{a' \in A} \hat{Q}^i(z_{t+1}, a')$ bzw. $\bar{q}_t^{i+1} = r_{t+1}$ für Übergänge in absorbierende Terminalzustände.

3. Überwachtes Lernen Training eines Funktionsapproximators auf den Trainingsdaten $\mathcal{P}^{i+1}$, um die $i+1$-te approximierte Q-Funktion $\hat{Q}^{i+1}$ zu erhalten.

4. Innere Schleife Erhöhung des Schleifenzählers $i \leftarrow i + 1$ und Fortfahren mit Schritt E2, solange der Bellman-Fehler über einer vorgegeben, sehr kleinen Schranke liegt. Andernfalls Rückgabe des berechneten Fixpunktes $\bar{Q}^k$.

F. Äußere Schleife Abbruch und Rückgabe der finalen Approximation $\bar{Q}^k$ und des zugehörigen Encoders $\text{ENC}(o; W^k)$, wenn Kriterium erreicht oder Ableiten einer neuen Explorationsstrategie π^k von der Approximation $\bar{Q}^k$ (z.B. mittels ϵ-greedy-Auswertung) und Fortfahren mit Schritt B.

Im allgemeinen Fall wird in Ermangelung jeglichen Wissens beim ersten Schleifendurchlauf eine initiale Strategie π^0 verwendet, die unabhängig von der Wahrnehmung zufällig Aktionen a aus A wählt. In diesem Fall kann auf einen initialen Encoder gänzlich verzichtet oder zum Beispiel die Identität verwendet werden.

Nach der Durchführung der Exploration in Schritt B können in Schritt C, neben den Ausgangsobservationen o_t aller gespeicherten Zustandsübergänge, zusätzlich die nachfolgenden Observationen o_{t+1} bei Übergängen in Terminalzustände für das Training der Autoencoder verwendet werden. Diese Beobachtungen am Ende einer Episode sind nicht gleichzeitig Ausgangsbeobachtung einer folgenden Transition und kämen sonst nicht in der Trainingsmenge vor. Die in Schritt C stattfindende Trainingsprozedur wird eingehend in Kapitel 5 erläutert.

In Schritt E läuft im Prinzip der unveränderte FQI-Algorithmus ab, nur dass er nicht auf den (hier unbekannten) Systemzuständen s oder den Observationen o, sondern auf den vom Encoder erzeugten Merkmalsvektoren z arbeitet. Die Auswirkungen der Verwendung der Merkmalsvektoren auf die Stabilität des FQI-Algorithmus werden eingehend in Kapitel 4 diskutiert. Der FQI-Algorithmus wird abgebrochen, wenn eine vorgegebene obere Schranke für den Bellman-Fehler $||\hat{Q}^i - \hat{Q}^{i-1}||_\infty$ unterschritten wird. In kürzester Pfad Problemen wird in dieser Abeit in der Regel eine Schranke in Höhe von 1/1000 der Kosten eines einzelnen Schrittes verwendet.

Die in Schritt E berechnete Q-Funktion $\bar{Q}^k : Z \times A \mapsto \mathbb{R}$ kann nun in Verbindung mit dem zugehörigen Encoder $\text{ENC}(\,\cdot\,; W^k)$ verwendet werden, um eine Strategie für das Observationen-MDP abzuleiten. Im Falle der gierigen Auswertung ergibt sich zum Beispiel $\pi(o) = \arg\max_{a \in A} \bar{Q}^k(z, a)$, wobei $z = \text{ENC}(o; W^k)$. Da die Wertfunktion $\bar{Q}^k$ ohne den Encoder für die von DFQ zu erlernende Steuerung des Observationen-MDPs wertlos ist, muss in Schritt F auch immer der passende Encoder mit zurückgegeben werden, um neue Observationen in den Merkmalsraum überführen zu können.

Noch offen ist die Auswahl eines geeigneten Funktionsapproximators innerhalb des FQI-Algorithmus, mit dessen Hilfe die Wertfunktion auf Basis der Trainingsbeispiele $\mathcal{P}$ approximiert wird. Bei der Auswahl sind die Stabilität und Dateneffizienz des Gesamtalgorithmus im Auge zu behalten, auf die der eingesetzte Approximator maßgeblichen Einfluss hat. Tatsächlich gestaltet sich die Auswahl trotz der vielen in Frage kommenden Verfahren gar nicht so einfach. Einige der zu beachtenden Randbedingungen werden bereits im folgenden Kapitel 4 aus theoretischer Sicht erarbeitet und weitere kommen im Kapitel 5 hinzu, bevor später in Kapitel 6 mit dem ClusterRL-Verfahren die Entwicklung und Abstimmung eines geeigneten Approximationsverfahrens beschrieben wird.

Nach der Durchführung des FQI-Algorithmus wird in Schritt F entschieden, ob mit einer weiteren Explorationsphase fortgefahren werden soll. Das einfachste Abbruchkri-

terium ist hier die Vorgabe einer maximalen Anzahl von Iterationen.

Der hier beschriebene Algorithmus ist zunächst wesentlich aufwändiger als "gewöhnliches" batch RL. Nicht nur, dass der FQI-Algorithmus während eines Lernexperiments mit Exploration viele Male ausgeführt wird, ein großer Aufwand ergibt sich insbesondere mit jedem Generationswechsel, dem Training eines neuen Encoders in Schritt C.

3.5 Effizienzsteigerung beim Generationswechsel

Mit jedem Training eines neuen Encoders in Schritt C von DFQ und der damit verbundenen Neuberechnung eines Merkmalsraums ändert sich die Bedeutung der Merkmalsvektoren grundlegend. Wertfunktionen und Strategien, die auf den alten Merkmalsvektoren basieren, können nicht mehr auf Merkmalsvektoren des neuen Raums angewendet werden und verlieren so ihre Bedeutung. Ein reines online Reinforcement Lernverfahren müsste nun aufgrund der veränderten Semantik alles bisher Erlernte verwerfen und von vorne beginnen – was unweigerlich zu einer Vervielfachung der insgesamt notwendigen Interaktionen mit dem System führen würde.

Im Folgenden werden einige – zum Teil heuristische – Ansätze disktutiert, die am theoretischen Ablauf von DFQ nichts wesentliches verändern, mit deren Hilfe die Effizienz beim Generationswechsel in der Paxis aber zum Teil deutlich gesteigert werden kann. Den ersten drei Ansätzen ist gemein, dass sie zwingend den Einsatz von speicherbasierten batch RL-Verfahren erfordern und online Verfahren in der inneren Schleife ausschließen.

3.5.1 Sofortige Neuberechnung einer Wertfunktion

Bei der Anwendung eines reinen online Verfahrens in der inneren Schleife hätte man bei einem Generationswechsel nur die aktuelle Schätzung der Wertfunktion zur Hand, welche aufgrund des veränderten Merkmalsraums ihre Bedeutung verliert. In DFQ wird dagegen ein batch Ansatz verwendet. Hier werden neben der approximierten Wertfunktion auch alle jemals beobachteten Transitionen zwischengespeichert. Werden diese Transitionen nicht nur im Merkmalsraum als Übergänge $(z_t, a_t, r_{t+1}, z_{t+1})$ von einem Merkmalsvektor z_t zu einem anderen Merkmalsvektor z_{t+1} gespeichert, sondern parallel dazu auch als Übergänge $(o_t, a_t, r_{t+1}, o_{t+1})$ im Observationsraum vorgehalten, ergibt sich die Möglichkeit, auch nach einem Generationswechsel direkt mit einer guten Strategie fortzufahren. So kann der neue Encoder $\mathrm{ENC}(\,\cdot\,; W^{k+1})$ dazu verwendet werden, die Menge der Übergänge direkt vom Observationsraum in den neuen Merkmalsraum Z^{k+1} zu übertragen. Zu diesem Zweck werden zunächst alle Transitionstupel $(o_t, a_t, r_{t+1}, o_{t+1}) \in \mathcal{F}_o$ mit Hilfe des neuen Encoders nach $(z_t^{k+1}, a_t, r_{t+1}, z_{t+1}^{k+1})$ übersetzt, wobei $z_t^{k+1} = \mathrm{ENC}(o_t; W^{k+1})$ (entsprechend Schritt D von DFQ). Auf Basis dieser Übergänge kann dann sogleich eine neue Wertfunktion für den neuen Merkmalsraum berechnet werden, ohne überhaupt mit dem System interagieren zu müssen.

Bei DFQ befindet sich so gesehen alles jemals erworbene Wissen über das System in den gesammelten Übergangsdaten und kann jederzeit durch Anwendung eines batch

RL-Verfahrens extrahiert und in Form einer Wertfunktion oder Strategie repräsentiert werden. Tatsächlich steckt in der neuen Wertfunktion im Prinzip das gleiche Wissen wie in der alten; ein Unterschied entsteht lediglich durch die bessere (oder schlechtere) Eignung der erlernten Einbettung.

Die hier beschriebene Möglichkeit wurde bereits in den DFQ-Algorithmus eingearbeitet (Schritt D; Neuberechnung aller Transitionen im Merkmalsraum) und wird in allen Experimenten verwendet. Sie ist zudem das wesentliche Argument für die ausschließliche Verwendung von batch Verfahren in DFQ und den Ausschluss von online Verfahren.

3.5.2 "Übersetzen" der Wertfunktion

Der Vorgang der vollständigen Neuberechnung der Wertfunktion wird mit längerer Trainingsdauer und anwachsender Datenmenge immer rechenaufwändiger, da die Approximation nach jeder Änderung in zahlreichen Iterationen über die wachsende Datenmenge ausgehend von einer initialen Approximation mit $\hat{Q}^0(z,a) = \bar{q}^0$ für alle $(z,a) \in Z \times A$ von Grund auf neu berechnet werden muss (siehe DFQ Schritt E.1). In der Folge wird beschrieben, wie die in der aktuellen Wertfunktion enthaltenen Kostenschätzungen auf den neuen Merkmalsraum übertragen werden können, um beim dynamischen Programmieren nicht von vorn beginnen zu müssen. Die entscheidende Idee besteht darin, einen zu E.2 ähnlichen DP-Schritt durchzuführen, bei dem die erwarteten Kosten der Nachfolgeobservationen o_{t+1} zwar aus der *alten* Q-Funktion unter Anwendung des *alten* Encoders ausgelesen werden, bei dem dann aber für den neu berechneten Zielwert des Observations-Aktionspaars (o_t, a_t) ein Trainingsbeispiel im *neuen* Merkmalsraum erzeugt wird:

Sei Z^k der alte Merkmalsraum vor dem Training eines neuen Encoders $\mathrm{ENC}(\,\cdot\,; W^{k+1})$. Sei $\bar{Q}^k : Z^k \times A \mapsto \mathbb{R}$ eine Approximation der Q-Funktion in diesem alten Merkmalsraum. Um auf Basis der alten Q-Funktion $\bar{Q}^k$ eine Q-Funktion $\bar{Q}^{k+1} : Z^{k+1} \times A \mapsto \mathbb{R}$ für den neuen Merkmalsraum Z^{k+1} zu berechnen, wird zunächst eine Trainingsmenge $\mathcal{P} = \{(z_t^{k+1}, a_t; \bar{q}_t) | t = 1, ..., p\}$ für alle vorhandenen Transitionen $\mathcal{F}_O = \{(o_t, a_t, r_{t+1}, o_{t+1}) | t = 1, \ldots, p\}$ durch Vollziehen eines leicht gegenüber Schritt E.2 abgewandelten DP-Schrittes berechnet. Das t-te Tupel $(z_t^{k+1}, a_t; \bar{q}_t)$ der Trainingsmenge $\mathcal{P}$ besteht dabei aus dem zur Observation o_t gehörenden Merkmalsvektor $z_t^{k+1} = \mathrm{ENC}(o_t; W^{k+1})$ im *neuen* Merkmalsraum, der zum Zeitpunkt t gewählten Aktion $a_t \in A$ und dem neuberechneten Q-Wert

$$\bar{q}_t = r_{t+1} + \gamma \max_{a' \in A} \bar{Q}^k(z_{t+1}^k, a') \ ,$$

wobei $z_{t+1}^k = \mathrm{ENC}(o_{t+1}; W^k)$ den Merkmalsvektor zur Observation o_{t+1} unter Anwendung des *alten* Encoders bezeichnet. Die so generierte Trainingsmenge wird dann verwendet, um die Approximation $\bar{Q}^{k+1}$ entsprechend Schritt E.3 durch überwachtes Lernen zu berechnen.

In Kapitel 4 wird gezeigt, dass der FQI-Algorithmus – wie hier beschrieben – von jeder beliebigen Ausgangsfunktion $\hat{Q}^0$ gestartet werden kann und immer zum gleichen

Fixpunkt konvergiert. Der Start mit der übersetzten Wertfunktion $\bar{Q}^{k+1}$ kann also nicht schaden. Die damit verbundene Hoffnung ist es, dass der FQI Algorithmus von der übertragenen Wertfunktion aus weniger Iterationen bis zur Konvergenz benötigt, als wenn er von einer zufällig initialisierten Wertfunktion aus startet. Besonders interessant ist diese Methode für die Anwendung von Verfahren wie NFQ, in denen das Training des Funktionsapproximators selbst einen relativ hohen Rechenaufwand hat und wo deshalb zwischen den Explorationsphasen prinzipiell nur wenige (1,2) DP-Schritte [130, 133] ausgeführt werden und der "Verlust" einer fortgeschrittenen Wertfunktion somit relativ teuer wäre.

3.5.3 Speichern der Transitionen im Merkmalsraum

Auch bei Anwendung der eben beschriebenen Methode bleibt eines unverändert: Die beobachteten Transitionen müssen alle im Observationsraum gespeichert werden. Das kann bei einer Vielzahl von hochauflösenden Bildern sehr speicheraufwändig werden. Bei der Speicherausstattung heutiger Rechner stellt dies meist kein ernstes Problem dar und im Rahmen dieser Arbeit wurde die Observationen immer vollständig gespeichert. Stellt der Speicherbedarf aber ein Kritierium dar, ist es möglich, das Abspeichern aller beobachteten Bilder teilweise zu umgehen, sofern das Merkmalsextraktionsverfahren wie im Falle der tiefen Autoencoder nicht nur den Encoder, sondern auch einen zugehörigen Decoder $o' = \mathrm{DEC}(z; W^k)$ liefert. In diesem Fall können die Bilder, die nicht für das Training eines guten Encoders benötigt werden, mit dem aktuell zur Verfügung stehenden Encoder $\mathrm{ENC}(\,\cdot\,; W^k)$ in den niedrigdimensionalen Merkmalsraum überführt und die Transitionen so speicherschonend abgespeichert werden. Die nicht benötigten Originalbilder werden in diesem Ansatz hingegen nicht zwischengespeichert, sondern sofort nach der Verarbeitung durch den Encoder "vergessen". Steht nun ein neuer Encoder $\mathrm{ENC}(\,\cdot\,; W^{k+1})$ zur Verfügung, können die Transitionen, für die keine Bilder, sondern nur Merkmalsvektoren vorhanden sind, durch einen einfachen Trick in den neuen Merkmalsraum übertragen werden. Dazu werden die Merkmalsvektoren z^k über den alten Decoder zurück in den Observationsraum und von dort in den neuen Merkmalsraum übertragen:

$$z^{k+1} = \mathrm{ENC}\left(\mathrm{DEC}(z^k; W^k); W^{k+1}\right) .$$

Aufgrund des unweigerlich entstehenden Rekonstruktionsfehlers zwischen nicht gespeicherter Observation o und deren Rekonstruktion $o' = \mathrm{DEC}(\mathrm{ENC}(o; W^k); W^k)$ entsteht bei der Übertragung der Transitionen mit diesem Verfahren allerdings ein Fehler, der sich auf zukünftige Kostenschätzungen auswirkt. In der Praxis ist es daher sinnvoll, die so übertragenen Transitionen zwar zur initialen Schätzung der Wertfunktion im neu konstruierten Merkmalsraum zu verwenden, mit Verfügbarwerden neuer Beobachtungen dann aber nach und nach zu ersetzen. Insbesondere mehrere Encodergenerationen alte Transitionen, die entsprechend oft von einem zum anderen Merkmalsraum übertragen wurden, können sonst zu einem zu großen, bleibenden Fehler beim dynamischen Programmieren führen.

3.5.4 Vermeidung von Generationswechseln

Für den Aspekt der Semantikänderung der Merkmalsvektoren und des drohenden Verlusts erworbenen Wissens konnten drei Strategien vorgestellt werden, die helfen, sowohl den zusätzlichen Interaktionsaufwand als auch den Rechenaufwand niedrig zu halten. Beobachtete Transitionen und erlernte Wertfunktionen können in den neuen Raum transferiert werden. Was allerdings als Problem bestehen bleibt, ist der hohe Rechenzeitaufwand, der mit dem Training eines tiefen Autoencoders selbst verbunden ist. Während der FQI-Algorithmus je nach Approximator auch für große Datenmengen innerhalb von Sekunden zu einem Ergebnis kommen kann, dauert das Training eines tiefen Autoencoders schon einmal mehrere Stunden (siehe auch Abschnitt 5.2.2). Die beste Strategie, den Rechenaufwand zu senken, besteht demnach darin, den Generationswechsel selbst wo immer möglich zu vermeiden.

Selteneres Nachtraining Prinzipiell macht es keinen Sinn, den Merkmalsraum jedes Mal sofort nach Erhalt von einigen wenigen neuen Observationen zu verändern, wenn dem damit verbundenen hohen Rechenaufwand gerade im späteren Lernverlauf nur geringe zu erwartende Verbesserungen gegenüberstehen. Als guter Kompromiss zwischen zu jedem Zeitpunkt bestmöglicher Einbettung und möglichst geringem Rechenaufwand hat sich die im Folgenden durchweg eingesetzte Strategie erwiesen, die Berechnung einer neuen Einbettung in Schritt 1a jeweils nach einer Verdoppelung der zur Verfügung stehenden Daten anzustoßen und ansonsten zu überspringen.

Einbringen von Vorwissen Eine weitere Technik zur Vermeidung von übermäßig vielen Generationswechseln gerade zu Beginn des Lernvorgangs kann das Einbringen von Vorwissen sein. Die initiale Explorationsstrategie π^0 und der initiale Encoder $\mathrm{ENC}(\,\cdot\,; W^0)$ dienen dazu, die allerersten Übergänge vom System einzusammeln, mit deren Hilfe dann im DFQ-Algorithmus eine erste Einbettung und Wertfunktion erlernt werden kann. Wird hier statt einer Zufallsexploration eine bessere, auf Vorwissen basierende Explorationsstrategie eingebracht oder wird mit Hilfe von Vorwissen beim Start des eigentlichen Lernvorgangs ein Satz von "geeigneten" Transitionen bereitgestellt, kann dies den Lernvorgang deutlich beschleunigen und die Anzahl der nötigen Generationswechsel senken. In der Praxis kann dies zum Beispiel heißen, zunächst einige zielführende Trajektorien von einem vorhandenen, aber nicht optimalen Kontroller einzusammeln um, erst dann DFQ mit diesen Daten zu starten.

Herauslösen des Encodertrainings In einigen Fällen kann es sinnvoll sein, ein System mit einer einfachen, sofort verfügbaren Strategie zu explorieren, einen Encoder zu trainieren und dann erst mit dem DFQ-Algorithmus zu starten, ohne den Encoder in der Folge noch zu ändern. Interessant ist diese Methode insbesondere bei Problemstellungen, bei denen die Dynamik eine Rolle spielt. Ein solches Beispiel wird in Kapitel 7 mit der Carrerabahn untersucht. Oftmals ist es in solchen Lernproblemen relativ leicht, eine Menge repräsentativer Bilder zu gewinnen, aber schwierig, die Grenzbereiche der

Dynamik zu explorieren. Das Herauslösen der Exploration führt in diesen Problemstellungen dazu, dass überhaupt keine Generationswechsel mehr auftreten und von Beginn bis Ende der eigentlichen Lernphase ausschließlich mit einem gleichbleibenden Merkmalsraum gearbeitet wird. Die Dynamik des Systems wird aber dennoch im ansonsten unveränderten Ablauf von DFQ mit immer besser werdenden Strategien exploriert.

Offline DFQ Im Extremfall kann DFQ auch als reiner offline Algorithmus ganz ohne eigene Exploration verwendet werden. Dem Algorithmus wird dann eine Menge von Transitionen übergeben, auf denen gleich ein Encoder berechnet wird. Die Exploration in Schritt B entfällt und es wird in F nach der ersten Iteration abgebrochen.

3.6 Zusammenfassung

In diesem Kapitel wurde der Rahmen geschaffen, visuomotorische Lernprobleme direkt auf Basis der Bilder zu lösen. Das Problem der hohen Dimensionalität der Eingabedaten wurde durch die Integration einer Methode zur automatischen Konstruktion von Merkmalsräumen in den batch RL-Ablauf angegangen. Das Resultat war neben einem allgemeinen, zweistufigen Rahmenablauf für Verfahren mit und ohne Exploration der DFQ-Algorithmus, in dem Ernsts Fitted Q-Iteration und das Training tiefer Autoencoder auf effiziente Weise kombiniert wurden.

Ein besonderes Problem der semi-online Varianten mit Exploration stellen die notwendigen Generationswechsel von einem alten zu einem neuen Merkmalsraum dar. Aber durch die bei batch Verfahren gegebene Möglichkeit zur sofortigen Neuberechnung einer Wertfunktion auf Basis der gespeicherten Übergänge sind diese Generationswechsel des Merkmalsraums in DFQ praktisch kostenfrei (Dateneffizienz, nicht Rechenzeit) möglich und führen zu keinem automatischen Anstieg des Interaktionsaufwands. Dies ist ein zentrales Argument für die Verbindung tiefen Lernens mit batch RL und wäre so mit den althergebrachten online Verfahren wie dem Q-Lernen nicht möglich. Darüber hinaus wurden weitere Strategien und Heuristiken entwickelt, die situativ zur Steigerung der Effizienz eingesetzt werden können. Insbesondere das Übersetzen der Wertfunktionen von einem in den anderen Zustandsraum und die Ideen zur Verminderung der Anzahl nötiger Generationswechsel sind Methoden, die für viele praktische Aufgabenstellungen interessant sind und zum Einsatz kommen können, ohne die Qualität des Lernergebnisses negativ beinflussen zu müssen.

In diesem Kapitel noch nicht besprochen wurden die praktische Realisierung des Trainings des Autoencoders innerhalb von DFQ und die Auswahl eines für die Anwendung auf automatisch erzeugte Merkmalsräume geeigneten Funktionsapproximators. Bevor diese beiden zentralen Module in den Kapiteln 5 und 6 besprochen werden, werden im folgenden Kapitel zunächst die formalen Eigenschaften des vorgeschlagenen Algorithmus näher beleuchtet. Diese Diskussion liefert weitere wichtige Gesichtspunkte für die Realisierung und Analyse der Konstruktion der Merkmalsräume und für die Auswahl des Funktionsapproximators.

4

Zur Konvergenz und Konsistenz des DFQ-Algorithmus

In diesem Kapitel werden zunächst die bestehenden theoretischen Resultate ausgeführt, auf denen der DFQ-Algorithmus als batch RL-Ansatz gründet. Die Konvergenzbeweise werden ausführlich wiedergegeben, da in der Literatur bisher keine umfassende Darstellung und Diskussion der unterschiedlichen Resultate existiert und insbesondere die konstruktiven Beweise wichtige Anhaltspunkte für den praktischen Umgang und die Probleme im Einsatz von approximativen Reinforcement Lernalgorithmen liefern, die wie DFQ auf der approximativen Wertiteration [51] bzw. dem kernelbasierten approximativen dynamischen Programmieren (KADP) [111–113] gründen. Über die Etablierung des von Ernst entwickelten FQI-Verfahrens [39] als einer Variante des KADPs wird in Abschnitt 4.2 eine Brücke von der Besprechung der theoretischen Grundlagen hin zur Analyse des neuen DFQ-Verfahrens geschlagen, in dem FQI eine zentrale Rolle spielt. Auf der so gelegten Basis werden schließlich zwei wesentliche neue Aussagen getroffen:

1. Ein Approximationsschema, das auf die Approximation einer Funktion $f : Z \mapsto \mathbb{R}$ im Merkmalsraum angewendet wird und bestimmten, erstmals von Gordon formulierten Kriterien (nicht-expansiv, "Averager") [51, 53] genügt, hat eine direkte Entsprechung im Observationsraum.

2. Aus 1. lässt sich die Gültigkeit einiger der Ergebnisse von Gordon und Ormoneit et al. [111–113] für DFQ folgern. Insbesondere wird der Nachweis erbracht, dass DFQ in der inneren Schleife für diskontierte Probleme zu einem eindeutigen Fixpunkt konvergiert, trotz der nichtlinearen Abbildung der Observationen in einen Merkmalsraum und des damit verbundenen, möglichen Verlusts der Markov-Eigenschaft.

Darüber hinaus werden mit der Qualität der gefundenen Lösung und der Konsistenz des Ergebnisses über mehrere Iterationen der äußeren Schleife hinweg weitere wichtige Eigenschaften des DFQ-Algorithmus diskutiert. Dabei werden auch praxisrelevante Lücken identifiziert, die Anlass geben zur Entwicklung eines geeigneten Funktionsap-

proximators und zu ausführlichen empirischen Untersuchungen der Merkmalsräume in den folgenden Kapiteln.

4.1 Grundlagen

Die formale Basis des heutigen batch Reinforcement Lernens hat ihren Ursprung in Publikationen von Geoffrey J. Gordon aus dem Jahr 1995 [51–53]. In [51] bzw. ausführlicher in [53] hat Gordon zunächst die Konvergenz eines approximativen Wertiterationsverfahrens (modellbasiert, siehe Abschnitt 2.2.4, Gleichung (2.7) und Abschnitt 2.2.7) auf einer festen Menge von Stützstellen für eine bestimmte Klasse von Funktionsapproximatoren nachgewiesen. Gleichzeitig wurde eine obere Schranke für den Fehler der Approximation gegenüber der gesuchten optimalen Wertfunktion bestimmt.

4.1.1 Konvergenz der approximativen Wertiteration

Der klassische Beweis der Konvergenz des (exakten) Wertiterationsverfahrens zu einer eindeutigen Lösung basiert auf dem Nachweis der Kontraktionseigenschaft des Aktualisierungsoperators und der darauf folgenden, direkten Anwendung des Fixpunktsatzes von Banach (bewiesen 1922 von Stephan Banach, bei Gordon in [53, Theorem 2.1], anschließend an mehreren Stellen verwendet). Das Fixpunkttheorem von Banach beinhaltet die Existenz und Eindeutigkeit der Lösung und gleichzeitig eine Konvergenzrate und Fehlerabschätzung.

Gordons Konvergenzbeweis für die approximative Wertiteration mit dem "Fitted Value Iteration"-Verfahren (FVI) [51, 53] gliedert sich in drei wesentliche Schritte: Zunächst wird eine Klasse von Funktionsapproximatoren definiert, die eine Funktion durch eine gewichtete Mittelung der an endlich vielen Stützstellen bestimmten Funktionswerte annähern (die sogenannten "Averager"). Dann wird nachgewiesen, dass die von diesem Funktionsapproximationsschema realisierten Abbildungen – von Zielfunktionen auf approximierte Funktionen – Kontraktionen (bzw. nicht-Expansionen) sind. Mit Hilfe dieses Ergebnisses lässt sich über die Feststellung, dass Verknüpfungen von Kontraktionen (bzw. nicht-Expansionen) diese Eigenschaft behalten, das Konvergieren zu einem eindeutigen Fixpunkt in direkter Anlehnung an den klassischen Beweis nachweisen. Im Folgenden werden diese zentralen Aussagen näher ausgeführt.

Definition 1. Nach Gordon [53] ist ein "Funktionsapproximationsschema" A genau dann ein Averager, wenn jeder approximierte Funktionswert ein gewichteter Durchschnitt von beliebig vielen Zielwerten (der Trainingsbeispiele) und beliebig vielen vordefinierten Konstanten ist. Dabei dürfen die Gewichte zwar von den Urbildern X in den Trainingsbeispielen, nicht aber von den Zielwerten Y abhängen. Für eine feste Menge von p Trainingsbeispielen $\mathcal{F} = \{(x_i, y_i)|i = 1, 2, \ldots, p\}$ mit $\bigcup_{i=1}^{p}\{x_i\} = X$ und $\bigcup_{i=1}^{p}\{y_i\} = Y$ berechnet sich der approximierte Funktionswert $\hat{y}_i$ an der Stelle x_i konkret nach

$$\hat{y}_i = \beta_i k_i + \sum_{j=1}^{p} \beta_{ij} y_j \ , \tag{4.1}$$

wobei die Gewichte β_i und β_{ij} genau wie k_i nicht-negative, reelle Zahlen sind und gilt:

$$\beta_i + \sum_j \beta_{ij} = 1 \; . \tag{4.2}$$

Unter diese Definition fallen viele verschiedene Typen von Funktionsapproximatoren, die sich in der zunächst nicht näher spezifizierten Art der Berechnung der Gewichte unterscheiden. Dazu gehören zum Beispiel einfache Diskretisierungen des Definitionsbereichs auf Basis regulärer und irregulärer Partitionierungen – auch auf Basis von Bäumen – mit konstanten Funktionswerten in den einzelnen Zellen, linear-interpolierende reguläre Gitter, k-nächste Nachbarn und kernelbasierte Mittelung. Lineare Regression, Neuronale Netze und auch CMACs sind keine Averager entsprechend dieser Definition [51, Abschnitt 3].

Definition 2. Sei $f : V \mapsto \mathbb{R}$ die zu approximierende Zielfunktion vom Vektorraum V nach $\mathbb{R}$. Für jede beliebige Menge von Trainingsbeispielen $\mathcal{F} = \{(x_i; y_i) | i = 1, 2, \ldots, p\}$ mit $y_i = f(x_i)$ für alle $(x_i; y_i) \in \mathcal{F}$ erzeugt das Funktionsapproximationsschema A eine approximierte Funktion (Approximation) $\hat{f} : V \mapsto \mathbb{R}$. Entsprechend [51] sei M_A nun definiert als die mit A assoziierte Abbildung auf dem Funktionenraum aller Funktionen $V \mapsto \mathbb{R}$, die jede Zielfunktion f auf die resultierende Approximation $\hat{f}$ bei Anwendung des Funktionsapproximationsschemas A auf eine Menge von Traininsgbeispielen $(x; f(x))$ abbildet.

Der entscheidende Schritt ist nun nachzuweisen, dass die mit einem Averager assoziierte Abbildung M_A eine nicht-Expansion ist.

Satz 1. *Die mit einem beliebigen Averager A assoziierte Abbildung M_A ist eine nicht-Expansion in Max-Norm $||f(x)||_\infty = \sup_x |f(x)|$.*

Beweis nach [53]. Sei $X \subset V$ eine Menge von zur Approximation verwendeten Stützstellen (die x_i in $\mathcal{F}$) und Y und Z zugehörigen Zielwerte zweier unterschiedlicher Zielfunktionen f und f' an diesen Stützstellen. Bezeichne $|| \cdot ||_i$ im Folgenden den Abstand am Element i, dann gilt an einer beliebigen Stützstelle $x_i \in X$:

$$\begin{aligned}
| \, ||M_A(Y) - M_A(Z)||_i \, | &= |(\beta_i k_i + \sum_j \beta_{ij} y_j) - (\beta_i k_i + \sum_j \beta_{ij} z_j)| && (4.3)\\
&= |\sum_j \beta_{ij}(y_j - z_j)| && (4.4)\\
&\leq \max_j (y_j - z_j) && (4.5)\\
&= ||Y - Z||_\infty && (4.6)
\end{aligned}$$

Die Gleichung (4.3) folgt aus der Definition der Averager, (4.5) gilt aufgrund der Bedingung (4.2). Aus dieser Ungleichung ergibt sich, dass alle Elemente des Vektors $[M_A(Y) - M_A(Z)]$ dem Betrag nach kleiner gleich $||Y - Z||_\infty$ sind, also die Differenz der approximierten Funktionen $\hat{f} = M_A(f)$ und $\hat{f}' = M_A(f')$ an allen Stützstellen kleiner

als die maximale Differenz zwischen den Zielfunktionen f und f' ist. In Max-Norm ergibt sich $||M_A(Y)-M_A(Z)||_\infty \leq ||Y-Z||_\infty$. Darüber hinaus kann die Differenz zwischen den Funktionen $\hat{f}$ und $\hat{f}'$ aufgrund der Konstruktion durch einen Averager auch zwischen den Stützstellen nicht größer sein als die maximale Differenz $||M_A(Y) - M_A(Z)||_\infty$ an den Stützstellen. Diese Feststellung ist wichtig, um auch eine Aussage über die Abfrage der approximierten Wertfunktionen zwischen den Stützstellen machen zu können. An einer beliebigen Stelle $x \in V$, die keine der Stützstellen X sein muss, gilt aber mit den für diese Stelle entsprechend Definition 1 festgelegten, individuellen Gewichten β und β_j analog zu (4.3-4.6):

$$
\begin{aligned}
|M_A(f)(x) - M_A(f')(x)\,| &= |(\beta k + \sum_j \beta_j y_j) - (\beta k + \sum_j \beta_j z_j)| && (4.7)\\
&= |\sum_j \beta_j (y_j - z_j)| && (4.8)\\
&\leq \max_j (y_j - z_j) && (4.9)\\
&= ||Y - Z||_\infty && (4.10)\\
&\leq ||f - f'||_\infty && (4.11)
\end{aligned}
$$

Die Ungleichung (4.11) folgt schlicht daraus, dass die maximale Differenz $||f - f'||_\infty$ nicht kleiner als die maximale Differenz $||Y - Z||_\infty$ der Funktionswerte Y und Z von f und f' sein kann. Also ist M_A eine nicht-Expansion unter der Max-Norm, sowohl bezüglich der Abbildung der Vektoren Y und Z als auch der Funktionen f und f'. □

Bevor nun auf Basis dieses Ergebnisses die Konvergenz der approximativen Wertiteration bewiesen werden kann, wird noch ein kleiner Hilfssatz über die Verknüpfungen von kontrahierenden bzw. nicht-expansiven Abbildungen benötigt.

Lemma 1. *Seien f und f' Kontraktionen mit Kontraktionsfaktoren λ und δ und g und g' nicht-Expansionen auf einem normierten Vektorraum. Dann gilt*

- *$f \circ g$ und $g \circ f$ sind Kontraktionen mit Kontraktionsfaktor λ,*
- *$f \circ f'$ ist eine Kontraktion mit Kontraktionsfaktor $\lambda\delta$ und*
- *$g \circ g'$ ist eine nicht-Expansion.*

Dieser Hilfssatz wurde aus [53] entnommen und bildet die Basis für nachfolgend wiedergegebenen Beweis der Konvergenz der approximativen Wertiteration für Diskontierungsfaktoren $\gamma < 1$.

Satz 2 (Konvergenz approximativer Wertiteration mit FVI)**.** *Sei H der Operator für die synchrone Aktualisierung im Wertiterationsverfahren (siehe Gleichung (2.7)) für ein gegebenes MDP M mit Diskontierungsfaktor $\gamma < 1$ und sei M_A eine zu einem Averager A gehörende Abbildung, dann ist $M_A \circ H$ eine Kontraktion in Max-Norm mit Faktor γ. Somit konvergiert die auf A basierende, approximative Wertiteration auf M mit der Rate γ in Max-Norm gegen einen eindeutigen Fixpunkt $\bar{V}$.*

Beweis nach [51]. Die Kontraktionseigenschaft in Max-Norm der Verknüpfung der nicht-Expansion M_A (Satz 1) mit der Kontraktion H mit Faktor γ ("value contraction" Theorem, wiedergegeben in [53]) ergibt sich aus Lemma 1. Die Konvergenz gegen einen eindeutigen Fixpunkt $\bar{V}$ folgt dann direkt aus dem Fixpunktsatz von Banach, auf dem auch der klassische Beweis des value contraction Theorems für die exakte Wertiteration basiert. □

Korollar 1. *Die Verwendung eines nicht-expansiven Funktionsapproximators ist ein notwendiges Kriterium für die Konvergenz der approximativen Wertiteration. Gibt es zwei Wertfunktionen f und f', für die gilt $||M_A(f) - M_A(f')||_\infty > ||f - f'||_\infty$, dann gibt es ein MDP M mit Operator H, das keinen eindeutigen Fixpunkt hat [51].*

Diese Aussagen erweitert Gordon so weit wie möglich auch auf stochastische kürzester Pfad Probleme mit $\gamma = 1$ und absorbierenden Zielzuständen $S_1 \subset S$. Der Nachweis der Konvergenz bzw. Stabilität in solchen Problemstellungen erweist sich als ungleich schwerer und hängt selbst für die exakte Wertiteration von einigen weiteren Bedingungen wie der Erreichbarkeit der absorbierenden Zielzustände und der Existenz von mit Sicherheit zielführenden Strategien ("proper policies") mit beschränkten Kosten ab [14–16]. Für die Kombination mit nicht-expansiven Funktionsapproximatoren ergeben sich Probleme, die eine Konvergenz zu einem eindeutigen Fixpunkt verhindern oder sogar Divergenz verursachen können. Gordon führt deshalb den Gedanken der Kompatibilität eines Approximators mit einem gegebenen MDP ein. Die Kompatibilität ist in der Praxis allerdings schwer zu überprüfen und setzt praktisch die vollständige Kenntnis und Analyse aller möglichen Systemübergänge voraus [53, Abschnitt 4]. Durch "inkompatible" Funktionsapproximatoren kann das ursprünglich die Bedingungen erfüllende MDP quasi auseinandergerissen werden, wodurch im approximativen Ansatz zirkuläre Wertaktualisierungen entstehen können, die hier aufgrund $\gamma = 1$ zu einer Divergenz führen [53, Abschnitt 4, Seite 10], während die Kosten für $\gamma < 1$ beschränkt blieben. Für den praktischen Umgang mit stochastischen kürzester Pfad Problemen ergeben sich daraus zwei Maßnahmen, um unabhängig von den Gewichten des eingesetzten Averagers theoretisch nachweisbare Stabilität sicherzustellen:

Sichere Diskontierung Der Diskontierungsfaktor kann minimal auf einen Wert knapp unter 1 abgesenkt werden, der bei geschickter Wahl keinen Einfluss auf die optimale Strategie hat, aber gleichzeitig formale Stabilität sicherstellt.

"Non-zero-weights"-Heuristik Es kann ein Averager verwendet werden, bei dem gar keine Gewichte mit $\beta_{ij} = 0$ auftauchen, sondern immer (auch minimal) über 0 liegen. So kann das von Gordon beschriebene "Auseinanderreißen" [53] des MDPs in vielen Fällen verhindert werden. Ein beliebiger Averager kann leicht entsprechend dieser Heuristik abgeändert werden, indem an jeder Stützstelle alle anderen Zielwerte mit einem sehr kleinen Gewicht $\beta_{ij} > 0$ hinzuaddiert werden. Dadurch wird zwar ein von der Größe der Gewichte abhängiger Bias eingeführt, aber auch der Betrag der approximierten Werte $\hat{y}_i$ nach oben beschränkt und so Divergenz effektiv verhindert.

4.1.2 Qualität der Lösung der approximativen Wertiteration

Hinsichtlich der Qualität der mittels approximativer Wertiteration gefundenen Lösung gibt Gordon eine obere Schranke für den Abstand des Fixpunktes $\bar{V}$ der approximativen Wertiteration zur (unbekannten) optimalen Wertfunktion V^* in Max-Norm an.

Satz 3. *Sei V^* die optimale Wertfunktion eines MDP M mit Diskontierungsfaktor $\gamma < 1$. Sei H der zugehörige synchrone Aktualisierungsoperator. Sei M_A die assoziierte Abbildung zum Averager A und sei V^A ein beliebiger Fixpunkt von M_A. Die Folge $(\hat{V}^i)$ approximierter Wertfunktionen unter wiederholter Anwendung des Operators $H \circ M_A$ konvergiert zu einem Fixpunkt $\bar{V}$, für dessen Abstand (Fehler) zur optimalen Wertfunktion folgende obere Schranke gilt:*

$$||V^* - \bar{V}||_\infty \leq \frac{2\gamma\epsilon}{1-\gamma} \,. \tag{4.12}$$

Diese Schranke hängt mit $\epsilon = ||V_A - V^*||_\infty$ vom Abstand des Fixpunktes V_A des Averagers zur optimalen Wertfunktion V^* ab.

Beweis. Der Beweis, der hier wiedergegeben wird, stammt aus [53]. Bezeichne $||\cdot||$ im Folgenden die Max-Norm.

$$\begin{aligned} ||\bar{V} - H(M_A(V^*))|| &= ||H(M_A(\bar{V})) - H(M_A(V^*))|| && (4.13)\\ &\leq \gamma||\bar{V} - V^*|| && (4.14) \end{aligned}$$

(4.13) folgt daraus, dass $\bar{V}$ Fixpunkt des Operators $H \circ M_A$ ist, (4.14) folgt aus Satz 2.

$$\begin{aligned} ||H(M_A(V^*)) - V^*|| &= ||H(M_A(V^*)) - H(V^*)|| && (4.15)\\ &\leq \gamma||M_A(V^*) - V^*|| && (4.16)\\ &\leq \gamma||M_A(V^*) - V^A|| + \gamma||V^A - V^*|| && (4.17)\\ &= \gamma||M_A(V^*) - M_A(V^A)|| + \gamma||V^A - V^*|| && (4.18)\\ &\leq \gamma||V^* - V^A|| + \gamma||V^A - V^*|| && (4.19) \end{aligned}$$

(4.15) folgt aus $H(V^*) = V^*$, weil V^* Fixpunkt von H, (4.16) folgt aus der Kontraktionseigenschaft von H. (4.17) ist die Anwendung der Dreiecksungleichung auf (4.16). (4.18) gilt, da für den Fixpunkt V^A von M_A per Definition $M_A(V^A) = V^A$. Aus M_A nicht-Expansion folgt schließlich (4.19). Auf Basis dieser Vorarbeiten kann man sich nun an die Abschätzung des Fehlers $||\bar{V} - V^*||$ machen.

$$\begin{aligned} ||\bar{V} - V^*|| &\leq ||\bar{V} - H(M_A(V^*))|| + ||\, H(M_A(V^*)) - V^*|| && (4.20)\\ &\leq \gamma||\bar{V} - V^*|| + 2\gamma||V^* - V^A|| && (4.21)\\ \Leftrightarrow (1-\gamma)||\bar{V} - V^*|| &\leq 2\gamma||V^* - V^A|| && (4.22)\\ \Leftrightarrow ||\bar{V} - V^*|| &\leq \frac{2\gamma\epsilon}{(1-\gamma)} && (4.23) \end{aligned}$$

(4.20) ist eine Anwendung der Dreiecksungleichung. (4.21) folgt aus den vorhergehenden Ungleichungen (4.13–4.14) und (4.15–4.19). Das zu beweisende Resultat (4.23) ergibt sich durch Äquivalenzumformungen, wobei $1 - \gamma > 0$. □

Diese Schranke bezieht sich immer auf den Fehler nach letztmaliger Anwendung des Operators H. Üblicherweise wird aber von den Lernalgorithmen die Approximation $\hat{V}^\infty = M_A(\bar{V})$ des Fixpunktes $\bar{V}$ zurückgegeben.

Korollar 2. *Als Schranke für den Fehler der Approximation $\hat{V}^\infty$ ergibt sich durch eine weitere Anwendung des Approximators M_A auf den Fixpunkt $\bar{V}$ und bloßes nachrechnen:*

$$||V^* - M_A(\bar{V})||_\infty \leq 2\epsilon + \frac{2\gamma\epsilon}{1-\gamma} \tag{4.24}$$

Hinsichtlich dieser Schranken (4.12) und (4.24) sind insbesondere drei Dinge beachtenswert, die die Aussagekraft für viele praktische Problemstellungen stark einschränken und später auch eine Rolle in der Analyse von DFQ spielen:

- Über ϵ ist der Approximationsfehler proportional zur Differenz zwischen optimaler Wertfunktion und nahegelegenstem Fixpunkt V^A des eingesetzten Averagers. Für eine einfache Diskretisierung des Zustandsraums mit einem regulären Gitter lässt sich beispielhaft zeigen, dass hilfreiche Aussagen über den maximalen Fehler und insbesondere für eine Verringerung des Fehlers bei einer Verkleinerung der Zellgröße getroffen werden können [53]. Bedingungen sind die Lipschitz-Stetigkeit der optimalen Wertfunktion V^* und dass alle Zustände x in einer Gitterzelle den gleichen Funktionswert $V(x')$ des Zentrums x' der Zelle zugewiesen bekommen [25, 26].

  In der Praxis treten aber in praktisch jedem interessanten RL-Problem Diskontinuitäten auf – zum Beispiel zwischen euklidisch benachbarten, aber durch eine Wand getrennten Zuständen in einem Labyrinth. Eine solche Diskontinuität übt großen Einfluss auf den globalen Fehler der Approximation aus, wenn sie nicht exakt durch den Averager erfasst wird. Beträgt die maximale Diskontinuität innerhalb derselben Zelle eines Gitterapproximators zum Beispiel d, so gibt es keinen Fixpunkt V^A des Averagers, der in Max-Norm näher als $d/2$ an V^* liegt.

- Die Angabe der Schranke in Max-Norm spiegelt wider, dass sich ein großer, nur lokaler Fehler – wie zum Beispiel im vorhergehenden Beispiel aufgrund einer nicht gut erfassten Diskontinuität entstanden – auf den gesamten Zustandsraum auswirken kann. Es gibt keine engere Schranke für die Stellen, die nicht durch diesen Fehler beeinflusst werden und an denen die Approximation besser gelingt. Stelle man sich als Beispiel ein Labyrinth vor, bei dem Startzustand (maximale erwartete Kosten) und Zielzustand (minimale erwartete Kosten) durch eine Mauer getrennt direkt nebeneinander liegen. In diesem Fall läge eine Wertfunktion, die allen Zuständen die Hälfte der Entfernung zwischen Ziel und Startzustand als Kosten zuordnet, im Rahmen der Schranke (4.24), wenn die Mauer nicht exakt z.B. durch eine Zellgrenze eines Gitterapproximators erfasst werden würde. Der Informationsgehalt einer solchen Wertfunktion läge aber nahe bei Null. Für die Ableitung einer Strategie wäre sie völlig wertlos.

- Tatsächlich führt der Diskontierungsfaktor im Nenner sogar dazu, dass sich ein – auch kleiner – lokaler Fehler für größer werdende Diskontierungsfaktoren noch stärker auswirkt, so dass die Schranke im Grenzwert für $\gamma \to 1$ gegen unendlich geht und auch für kleine ϵ jegliche Aussagekraft verliert. Dies spiegelt wider, dass ohne eine weitere Kenntnis der Struktur der Zustandsübergänge davon auszugehen ist, dass sich ein lokaler Fehler an einer Stützstelle potenziell auf alle anderen Schätzungen im Rahmen des Diskontierungsfaktors auswirkt und sich mit den durch M_A eingeführten Approximationsfehlern aufsummiert. Ganz unabhängig vom konkreten Ansatz ist wichtig, dass sich ohne Kenntnis und genaue Analyse des Übergangsmodells der Einfluss kleiner, sehr lokaler Fehler auf alle andere Zustände in stochastischen kürzester Pfad Problemen mit $\gamma = 1$ praktisch nicht begrenzen lässt.

Gordon ging auch kurz auf die für DFQ interessante Anwendung seines Verfahrens auf das Erlernen von Q-Funktionen auf Basis beobachteter Zustandsübergänge ein. An Stelle der Verwendung des exakten Modells müssten die Übergangswahrscheinlichkeiten mittels stochastischer Approximation auf Basis der Stichprobe geschätzt werden [53, Abschnitt 5]. Die Praxistauglichkeit scheitert daran, dass die Aktualisierungen für die Gültigkeit von Gordons Theoremen an den festen Stützstellen vorgenommen werden müssen, aber in der Praxis wohl nur schwer ausreichend Übergänge gezogen werden können, die exakt an diesen Stützstellen liegen und hier ein exaktes Modell ersetzen könnten. Die Lösung für dieses Problem liefern die im nächsten Abschnitt beschriebenen stichprobenbasierten Verfahren.

4.1.3 Konvergenz des Lernens auf Basis einer Stichprobe

Um 2000 wurde der Averager-Ansatz von Ormoneit mit Sen [113] und Glynn [111, 112] erfolgreich auf das modellfreie Lernen übertragen. Ihr Ansatz basiert auf einer zentralen, neuen Idee: Der Vorgang der Approximation der Wertfunktion wird als stochastischer Prozess aufgefasst, mit dem Ziel, eine Schätzung des exakten Wertiterationsoperators auf Basis einer Stichprobe von Übergängen (daher auch als "sample-basiert" bezeichnet) vorzunehmen. Argumentiert wird dann über die Qualität der Schätzung $\hat{V}(s)$ für eine gegebene Stichprobe $\mathcal{F}$, bzw. über Erwartungswerte und Wahrscheinlichkeiten und nicht mehr über einen absoluten Fehler. Die Schätzung der tatsächlichen Übergangswahrscheinlichkeiten wird in ihrem Verfahren an Hand der beobachteten Übergangshäufigkeiten von ähnlichen Zuständen in der Stichprobe vorgenommen und durch eine kernelbaiserte Mittelung berechnet. An dieser Stelle findet sich die zentrale Idee von Gordon wieder; die vorgenommene Approximation ist ebenfalls nicht-expansiv. Aber anders als von Gordon werden von Ormoneit et al. mit dem Kernel-Based Averaging und den besonders geeigneten Gauß-Kerneln konkrete Methoden zur Gewichtung in der vorgenommenen Mittelung vorgeschlagen.

Allein durch diese leichte Veränderung des Blickwinkels – von der Schätzung der Kosten an festen Zustands-Aktionspaaren hin zur Schätzung der Kosten entlang tatsächlich beobachteter Transitionen und Mittelung über die beobachteten Übergänge für

ähnliche Zustands-Aktionspaare – wird das stichprobenbasierte Q-Lernen möglich.

Ormoneits Arbeiten werden an dieser Stelle aus zwei Gründen diskutiert. Zunächst bilden diese Überlegungen die theoretische Basis hinsichtlich der Konvergenz stichprobenbasierter batch RL-Ansätze und erfassen sowohl die Wertiteration auf Basis einer Stichprobe als auch das in DFQ eingesetzte Q-Lernen. Desweiteren liefern die Artikel auch über Gordons Resultate hinausgehende, formale Aussagen dazu, unter welchen Bedingungen der Abstand der gefundenen Lösung zur optimalen Wertfunktion (beliebig) verringert werden kann. Tatsächlich wird von Ormoneit nicht die Qualität einer einzelnen Schätzung (wie bisher bei Gordon) in den Vordergrund gestellt, sondern die Konsistenz, d.h. die Entwicklung der Schätzungsgüte bei Vergrößerung der Stichprobe. Dies ist insbesondere relevant für die äußere Schleife des DFQ-Algorithmus.

Ormoneit und Sens Vorschlag ist es, den im modellfreien Ansatz unbekannten, exakten DP-Operator H in der Bellman-Gleichung (2.9) mit $V^* = HV^*$ durch einen Zufallsoperator $\hat{H}$ auf Basis einer Stichprobe anzunähern.[1] Bevor die Konstruktion dieses Zufallsoperators näher erläutert und seine Konvergenz hergeleitet werden kann, werden einige grundlegende Definitionen und Beobachtungen bezüglich des Zusammenhangs zwischen Zustandswertfunktion und Q-Funktion benötigt.

Lemma 2. *Der zwischen optimaler Q-Funktion Q^* und optimaler Zustandswertfunktion V^* bestehende Zusammenhang lässt sich ausnutzen, um Gleichung (2.12) wie folgt auszudrücken:*

$$\begin{aligned} Q^*(s,a) &= E[r_{t+1} + \gamma \max_{a' \in A} Q^*(s_{t+1}, a') | s_t = s, a_t = a] && (4.25) \\ &= E[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \, . && (4.26) \end{aligned}$$

Gleichzeitig gilt für die optimale Zustandswertfunktion

$$V^*(s) = \max_{a \in A} Q^*(s,a). \quad (4.27)$$

Beweis. Folgt aus den Definitionen der optimalen Q-Funktion und Zustandswertfunktion und durch Einsetzen der Definitionen für die optimale Strategie (2.6) und (2.15). Weiterhin gilt $\max_{a \in A} Q^*(s,a) = Q^*(s, \arg\max_{a \in A} Q^*(s,a))$. □

Definition 3. Die optimale Aktionswertfunktion

$$Q_a^*(s) = E\left[r_{t+1} + \gamma Q^*\left(s_{t+1}, \pi^*(s_{t+1})\right) | s_t = s, a_t = a\right]$$

sei die Funktion, die die erwartete Belohnung angibt, wenn im Zustand $s_t = s$ die Aktion $a_t = a$ ausgewählt und ab dann die optimale Strategie verfolgt wird, also $Q_a^*(s) = Q^*(s,a)$.

[1] Die Bezeichnung von $\hat{H}$ als Zufallsoperator ("random-operator") wurde von Ormoneit et al. eingeführt, um deutlich zu machen, dass es sich nicht um den exakten Operator H, sondern nur um eine (ungenaue) Schätzung $\hat{H}$ der exakten Operation H handelt, die auf Basis einer Stichprobe vorgenommen wird und deshalb vom Zufall beeinflusst ist.

Diese Funktion Q_a^* wird später benötigt, um die Q-Funktion in einer separaten Funktion Q_a^* je Aktion $a \in A$ approximieren zu können.

Definition 4. Der Operator H_{dp}^a sei der Operator, der eine optimale Zustandswertfunktion V^* auf die zugehörige Aktionswertfunktion $Q_a^* = H_{dp}^a V^*$ mit

$$\begin{aligned} Q_a^*(s) &= H_{dp}^a V^*(s) \\ &= E[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \end{aligned} \tag{4.28}$$

abbildet. Folglich gilt für den Wert eines Zustandsaktionspaares (s, a) auch $Q^*(s, a) = Q_a^*(s) = H_{dp}^a V^*(s)$.

Der Operator H_{dp}^a führt wie der Operator H einen DP-Schritt durch (vgl. Abschnitt 2.2.4), dieser ist aber im Falle von H_{dp}^a quasi auf eine Aktion $a \in A$ beschränkt, so dass die Zustandswertfunktion auf eine "aktualisierte" Aktionswertfunktion Q_a abgebildet wird.

Korollar 3. *Sei die Q-Funktion Q definiert als die Menge der einzelnen Aktionswertfunktionen $Q = \{Q_a | \ a \in A\}$. Sei H_{max} der Maximumoperator über alle Aktionen $a \in A$ mit $H_{max}Q(s) = \max_{a \in A} Q(s, a) = H_{max}Q_a(s) = \max_{a \in A} Q_a(s)$. Dann ist $V = H_{max}Q$ und die Bellman-Gleichung $V^* = HV^*$ bzw. $V^* = H_{max}H_{dp}^a V^*$ kann unter Ausnutzung der Definition 3 wahlweise wie folgt geschrieben werden:*

$$V^* = HH_{max}Q^* = H_{max}H_{dp}^a H_{max}Q^* \tag{4.29}$$

$$Q^* = H_{dp}^a H_{max}Q^* \tag{4.30}$$

$$Q_a^* = H_{dp}^a H_{max}Q_a^* \tag{4.31}$$

Damit sind alle Werkzeuge beisammen, die von Ormoneit et al. verwendet werden, um die Q-Funktion nach Aktionen getrennt durch mehrere Aktionswertfunktionen repräsentieren und zwischen Zustandswertfunktionen, Aktionswertfunktionen und Q-Funktionen wechseln zu können. Ist eine dieser Funktionen bekannt, können die anderen mit Hilfe dieser Werkzeuge leicht abgeleitet werden. Der DP-Operator H in der Wertiteration lässt sich zudem durch die neuen Operatoren $H = H_{max}H_{dp}^a$ ersetzen, wobei H_{dp}^a genau genommen nicht einen Operator sondern verschiedene Operatoren für die Aktionen $a \in A$ bezeichnet. H_{dp}^a nimmt einen DP-Schritt hin zu einer Aktionswertfunktion vor. Soll die DP-Operation auf Aktionswertfunktionen angewendet werden, kommen $H_{dp}^a H_{max}$ (erzeugt Aktionswertfunktion) oder auch $H_{max}H_{dp}^a H_{max}$ (erzeugt Zustandswertfunktion) an Stelle von HH_{max} in (4.29) zum Einsatz.

Die Idee ist es nun, mit Hilfe dieser Werkzeuge, anders als bei Gordon, nicht den exakten DP-Operator H auf eine approximierte Wertfunktion $\hat{V}$ anzuwenden – denn dies würde die Kenntnis des Übergangsmodells T und R voraussetzen – sondern den exakten DP-Operator selbst auf Basis einer Stichprobe zu schätzen und diese Schätzung $\hat{H}$ an Stelle von H in approximativen Bellman-Gleichungen der Form $\hat{V} = \hat{H}\hat{V}$ zu verwenden. Ormoneit und Sen schlagen eine konkrete Konstruktion des Operators $\hat{H}_{dp}^a$ und einen RL-Algorithmus zur Lösung der approximierten Bellman-Gleichung

$$\hat{Q} = \hat{H}_{dp}^a H_{max}\hat{Q} \tag{4.32}$$

vor [113]. Auch wenn der Algorithmus auf der Q-Lernregel basiert, können von $\hat{Q}$ leicht zugehörige (approximierte) Zustandswertfunktionen und natürlich auch Strategien abgeleitet werden. Die Konstruktion des Operators und die Konvergenzeigenschaften des Algorithmus werden in der Folge näher erläutert.

Definition 5. Der approximative Erwartungswertoperator $\hat{H}^a_{dp}$ sei folgendermaßen für eine Stichprobe $\mathcal{F} = \{(s_t, a_t, r_{t+1}, s_{t+1}) |\ t = 1, \ldots, p\}$ definiert:

$$\hat{H}^a_{dp} V(s_0) = \sum_{(s,a,r,s') \in \mathcal{F}_a} k(s, s_0)[r + \gamma V(s')] \ , \tag{4.33}$$

wobei $\mathcal{F}_a \subset \mathcal{F}$ die Teilmenge der Übergänge $(s_t, a_t, r_{t+1}, s_{t+1}) \in \mathcal{F}$ ist, bei denen Aktion $a \in A$ ausgewählt wurde, also $a_t = a$. Der Term $\hat{H}^a_{dp} V(s)$ ist eine Approximation $\hat{Q}_a(s)$ der zu $V(s)$ gehörenden Aktionswertfunktion $Q_a(s)$.

Bei $k(s, s')$ handelt es sich um keine ganz gewöhnliche Kernelfunktion. Vielmehr handelt es sich um eine auf den Satz von Übergängen $\mathcal{F}_a$ speziell abgestimmte Gewichtsfunktion, für die die gleichen Bedingungen wie schon für Gordons Averager (siehe Definition 1) gelten: Die Gewichte, die den in $\mathcal{F}_a$ enthaltenen Transitionen zugewiesen werden, müssen positiv sein und sich zu eins summieren; also

$$k(s, s_0) \geq 0 \tag{4.34}$$

für alle $s_0 \in S$ und alle $(s, a, r, s') \in \mathcal{F}_a$, sowie

$$\sum_{(s,a,r,s') \in \mathcal{F}_a} k(s, s_0) = 1 \tag{4.35}$$

für alle $s_0 \in S$. Es gibt viele denkbare Möglichkeiten, einen solchen Kernel zu konstruieren. Im Zuge der Beweisführung (insbesondere hinsichtlich der Konsistenz) schlagen Ormoneit und Sen [113] die Verwendung eines "Mutterkernels" ϕ^+ in der folgenden Konstruktion vor.

Definition 6. Für die Stichprobe $\mathcal{F}_a$ sei die Gewichtsfunktion in der Folge für alle $s_0 \in S$ und alle $(s, a, r, s') \in \mathcal{F}_a$ definiert als

$$k_{\mathcal{F}_a, b}(s, s_0) = \ \phi^+ \left(\frac{||s - s_0||}{b} \right) \Bigg/ \sum_{(s_i, a_i, r_i, s'_i) \in \mathcal{F}_a} \phi^+ \left(\frac{||s_i - s_0||}{b} \right) . \tag{4.36}$$

Dabei sei ϕ^+ zum Beispiel eine univariate Gauß-Funktion und b ein Parameter zur Kontrolle der Bandbreite, dem effektiven Einzugsbereich des Kernels.

Die Idee hinter diesem Vorgehen lässt sich wie folgt interpretieren: In Gleichung (4.33) wird der Erwartungswert aus Gleichung (4.28) an Hand der Übergänge in der Stichprobe $\mathcal{F}_a$ geschätzt. Im Vergleich zu den diskreten MDPs ist es im kontinuierlichen Fall aber sehr unwahrscheinlich, dass der Zustand s nach endlicher Zeit oft genug exakt

getroffen wurde, so dass auf Basis der von ihm ausgehende Übergänge (s, a, r, s') in $\mathcal{F}$ eine exakte Schätzung der Übergangswahrscheinlichkeiten vorgenommen werden kann. Während Gordon dieses Problem in seiner Diskussion der Anwendung von FVI auf das Erlernen von Q-Funktionen in [53] nur benennt und nicht lösen kann (siehe auch Seite 60), bieten Ormoneit et al. die Verwendung umliegender Übergänge als Lösung an. Unter der Annahme einer in der Umgebung von s "glatten" Zustandswertfunktion $V(s)$ werden Übergänge aus der Umgebung von s zur Schätzung von $\hat{H}^a_{dp}V(s)$ als lokales, gewichtetes Mittel in Gleichung (4.36) herangezogen.

Gegenüber der stochastischen Approximation im gewöhnlichen Q-Lernen ist ein willkommener Vorteil, dass die Lernrate α wegfallen kann, da sich die Übergangswahrscheinlichkeiten direkt aus den in eine Zelle (bzw. in die Nachbarschaft) fallenden Übergängen $(s, a, r, s') \in \mathcal{F}_a$ durch einfache Mittelung abschätzen lassen. Aufgrund dieser Überlegungen ergibt sich folgender Algorithmus.

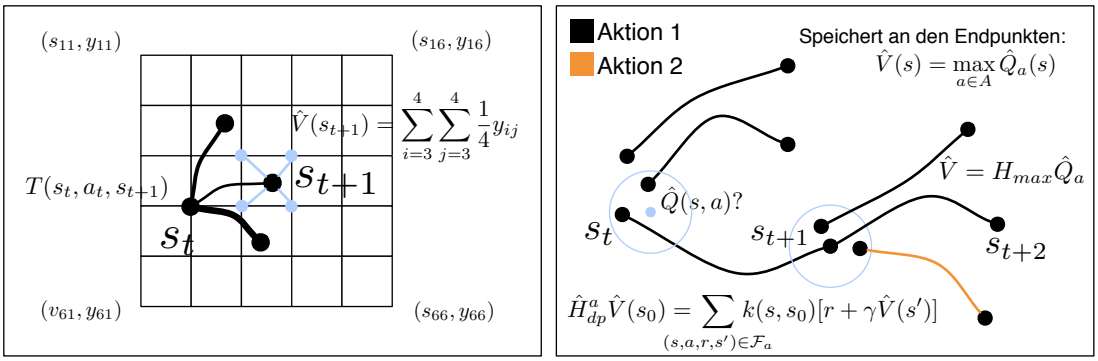


Abbildung 4.1: Grafische Darstellung der approximativen Wertiteration nach Gordon (links) und des kernelbasierten approximativen dynamischen Programmierens nach Ormoneit und Sen (rechts). Während bei Gordon die Wertfunktion an festen Stützstellen – hier an den Vertices eines Gitters – approximiert wird, wird sie bei Ormoneit und Sen an Hand der Daten in der Umgebung (blau) geschätzt.

Definition 7 (Kernel-based Approximate Dynamic Programming (KADP))**.** Ausgehend von einer initialen Schätzung der Zustandswertfunktion $\hat{V}^0$ wird ein Schritt des KADP-Verfahrens durch Lösung der Gleichung $\hat{V}^{i+1} = H_{max}\hat{H}^a_{dp}\hat{V}^i$ für die Übergänge (s, a, r, s') in der Stichprobe $\mathcal{F}$ vollzogen. $\hat{H}^a_{dp}$ wird dazu zur Berechnung der Aktionswertfunktionen $\hat{Q}^{i+1}_a$ auf Basis der Teilmengen $\mathcal{F}_a$ für alle Aktionen $a \in A$ entsprechend Gleichung (4.33) herangezogen, H_{max} bildet anschließend das Maximum über alle Aktionswertfunktionen. Es ergeben sich folgende Aktualisierungsregeln:

$$\hat{Q}^{i+1}_a(s_0) \leftarrow \sum_{(s,a,r,s') \in \mathcal{F}_a} k(s, s_0)[r + \gamma\hat{V}^i(s')] \tag{4.37}$$

$$\hat{V}^{i+1}(s_0) \leftarrow \max_{a \in A} \hat{Q}^{i+1}_a(s_0) \tag{4.38}$$

für die Aktionswertfuntion $\hat{Q}_a^{i+1}$ und die Zustandswertfunktion $\hat{V}^{i+1}$. Entscheidend ist die Beobachtung, dass $\hat{V}^i$ in der Berechnungsvorschrift (4.38) des KADP-Verfahrens niemals an den Anfangspunkten der Übergänge, sondern nur an den Endpunkten ausgewertet wird. Dementsprechend schlagen Ormoneit und Sen die Speicherung der Zustandswerte an den Endpunkten der Transitionen vor (siehe Abbildung 4.1). Die Zustandswerte an den Ausgangspunkten der Übergänge können, müssen aber nicht explizit repräsentiert werden, denn der Zustandswert an allen anderen s kann leicht durch Anwendung der Gleichung (4.33) nachträglich bestimmt werden. In diesem Sinne sei $H_{dp}^a V$ selbstapproximierend [113, 143].

Anmerkung: In [113] geben Ormoneit und Sen den Algorithmus mit Hilfe von Matrizen und Tensoren wieder. Ihre Schreibweise setzt aber voraus, dass je Aktion gleich viele Samples vorhanden sind und sie ist etwas ungenau in der Referenzierung der zu den (kontinuierlichen) Zuständen gehörenden Matrixeinträge. Daher wurde hier die klarere, algorithmische Darstellung gewählt.

Die Konvergenz von KADP lässt sich unter Zuhilfenahme von $|\sup f - \sup g| \leq \sup |f - g|$ für beliebige Funktionen $f, g : A \mapsto \mathbb{R}$ wie folgt beweisen.

Satz 4. *Im KADP-Verfahren konvergiert die durch mehrfache Anwendung der Aktualisierungsvorschriftem (4.37,4.38) gewonnene Folge $(\hat{V}^i)$ der approximierten Zustandswertfunktionen $\hat{V}^i$ ausgehend von einer beliebigen initialen Schätzung $\hat{V}^0$ auf diskontierten MDPs mit $\gamma < 1$ gegen einen eindeutigen Fixpunkt $\bar{V}$.*

Beweis nach Ormoneit und Sen [113, Anhang A.2]. Für die Differenz zwischen zwei verschiedenen Wertfunktionen V und V' ergibt sich nach Anwendung des approximativen DP-Operators $\hat{H}$ in Max-Norm

$$||\hat{H}V - \hat{H}V'||_\infty = \sup_{s \in S} \left| \max_{a \in A} \hat{H}_{dp}^a V(s) - \max_{a \in A} \hat{H}_{dp}^a V'(s) \right| \tag{4.39}$$

$$\leq \sup_{s \in S} \max_{a \in A} \left| \hat{H}_{dp}^a V(s) - \hat{H}_{dp}^a V'(s) \right| \tag{4.40}$$

$$= \sup_{s \in S} \max_{a \in A} \left| \sum_{(s_i, a_i, r_i, s_i') \in \mathcal{F}_a} k(s_i, s) \left[r + \gamma V(s') \right] - \right.$$

$$\left. \sum_{(s_i, a_i, r_i, s_i') \in \mathcal{F}_a} k(s_i, s) \left[r + \gamma V'(s') \right] \right| \tag{4.41}$$

$$= \sup_{s \in S} \max_{a \in A} \left| \sum_{(s_i, a_i, r_i, s_i') \in \mathcal{F}_a} k(s_i, s) \left(r + \gamma V(s') - r - \gamma V'(s') \right) \right| \tag{4.42}$$

$$= \gamma \sup_{s \in S} \max_{a \in A} \left| \sum_{(s_i, a_i, r_i, s_i') \in \mathcal{F}_a} k(s_i, s) \left(V(s') - V'(s') \right) \right| \tag{4.43}$$

$$\leq \gamma ||V - V'||_\infty \tag{4.44}$$

Gleichung (4.39) folgt aus der Definition von $\hat{H}$ als $H_{max}\hat{H}^a_{dp}$ (siehe auch Korollar 3). (4.41) folgt aus Definition 4.33. Da in der Differenz auf beiden Seiten über die gleiche Menge F_a summiert wird, folgt (4.42) und durch Streichen der sich aufhebenden sofortigen Belohnungen und durch anschließendes Herausziehen der Diskontierung ergibt sich (4.43). Gleichung (4.44) ergibt sich schon wie bei (4.5) aus der Definition der Gewichte $k(s_0, s)$ laut (4.35) (summieren zu 1). Die Konvergenz der KADP-Aktualisierung und die Eindeutigkeit des Fixpunktes ergeben sich damit direkt aus dem Fixpunktsatz von Banach. □

Eine wichtige Beobachtung ist, dass Ormoneit und Sen zwar mit (4.36) eine spezielle Kernelfunktion angeben, der Konvergenzbeweis aber nicht auf seinem Einsatz beruht und lediglich die Einhaltung der Normalisierungsbedingungen (4.34) und (4.35) verlangt. Unter diese Bedingunge fallen die gleichen Funktionsapproximatoren, die bei Gordon als "Averager" erfasst [39, 113] und somit prinzipiell in KADP eingesetzt werden können. Zu den kernelbasiert formulierbaren Funktionsapproximatoren gehören beispielsweise n-nächste-Nachbar-Regression, Gitterapproximatoren [97, 112] und Bäume [39, 112]. Die an den Kernel gestellten Bedingungen sind im Übrigen die gleichen wie in FQI (vgl. Abschnitt 4.2).

Neben diesem zentralen Beweis der Konvergenz für diskontierte MDPs gehen Ormoneit und Sen nur sehr kurz auf nicht diskontierte MDPs ein, ohne die von Gordon ausführlich diskutierten Probleme überhaupt zu erwähnen. Aufgrund der Verwandtschaft des Approximationsverfahrens gelten die für die approximative Wertiteration beobachteten Probleme mit inkompatiblen Funktionsapproximatoren eins-zu-eins für das KADP-Verfahren, so dass im Falle kürzester Pfad Probleme mit $\gamma = 1$ dieselbe Vorsicht wie zuvor bei der Auswahl der gewichtenden Kernelfunktionen geboten ist (siehe Diskussion und Maßnahmen auf Seite 57). Eine zentrale Rolle nimmt in den Ausführungen von Ormoneit und Sen die Konsistenz des Algorithmus ein.

4.1.4 Stochastische Konsistenz des Lernens auf Stichproben

KADP findet auf Basis einer Stichprobe $\mathcal{F}$ eine eindeutige Lösung der approximativen Bellman-Gleichung (4.32) für die Q-Funktion. Es verbleibt aber die Frage – wie schon in Abschnitt 4.1.2 – wie nah der mittels KADP gefundene Fixpunkt $\bar{Q} = \hat{H}^a_{dp}H_{max}\bar{Q}$ an der (unbekannten) Lösung Q^* der exakten Bellman-Gleichung (4.30) liegt.

Ormoneit und Sen verfolgen hier einen Ansatz, der zu einer wesentlich stärkeren Aussage als die reine Bestimmung einer oberen Schranke führt. Die Qualität der mittels KADP erzielten Schätzung $\bar{Q}$ der tatsächlichen optimalen Wertfunktion Q^* hängt bei KADP nicht nur von den Eigenschaften der gesuchten optimalen Wertfunktion und dem eingesetzten Kernel ab, sondern wird auch in ganz entscheidendem Maße von der tatsächlich in der Stichprobe enthaltenen Information in Form der gezogenen Übergänge beeinflusst. Gibt es in der Stichprobe zum Beispiel eine statistisch ungewöhnliche Häufung von eigentlich unwahrscheinlichen, aber besonders gewinnbringenden Übergängen aus einem Zustand oder wurde ein wichtiger Übergang in einen absorbierenden Terminalzustand (oder entlang eines "Flaschenhalses") zufällig oder aufgrund schlechter

Exploration nicht ein einziges Mal getroffen, kann der Fehler zwischen Approximation $\bar{Q}$ und tatsächlicher Wertfunktion Q^* fast beliebig groß werden – ganz unabhängig von der Kompatibilität der gewählten Gewichte.

Statt nun eine konkrete Schranke für den Fehler auf einer spezifischen Stichprobe anzugeben – praktisch ein unmögliches Unterfangen, ohne nicht schon vorab die optimale Wertfunktion und die Übergangsfunktionen genau zu kennen – nehmen Ormoneit und Sen in Kauf, dass die individuelle Schätzung beliebig von der optimalen Wertfunktion abweicht und beweisen dann aber die stochastische Konsistenz des Algorithmus. Wird die Größe der Stichprobe $\mathcal{F}$ kontinuierlich vergrößert, führt dies nicht nur zu einer Verringerung des erwarteten Fehlers der Schätzung $\bar{Q}$. Vielmehr konvergiert die Folge $(\bar{Q}^j)$ der Schätzungen $\bar{Q}^j$ für größer werdende Datenmengen sogar nachweislich gegen die optimale Q-Funktion Q^*, wenn gleichzeitig die Bandbreite der Kernel (4.36) in geeigneter Weise abgesenkt wird. Intuitiv kann man sich dies so vorstellen, dass die Wahrscheinlichkeit von Übergangshäufungen in der Stichprobe, die stark vom Modell T abweichen, genauso wie die Wahrscheinlichkeit, dass wichtige Übergänge in der Stichprobe fehlen, im Limes gegen 0 geht, wenn die Größe der Stichprobe kontinuierlich gegen unendlich erhöht wird. Lässt die größer werdende Stichprobe zudem eine (beliebige) Absenkung der Bandbreite zu, wird also die einflussausübende Umgebung immer kleiner, verringert sich auch der durch die Approximation eingeführte Fehler zunehmend [26, 53, 113].

Annahmen Bei der Beweisführung ist es nun erforderlich, dass im gewählten Zufallsoperator $\hat{H}^a_{dp}$ ein Kernel der Form (4.36) zum Einsatz kommt. Weitere wichtige Annahmen umfassen insbesondere die Stetigkeit der Belohnungsstruktur, der gesuchten Wertfunktion und der eingesetzten Kernel [113, Anhang A.1].

Definition 8 (Zulässige Verkleinerungsrate)**.** Eine Verkleinerungsrate $b(m)$ in Zusammenhang mit den in der Konstruktion von $\hat{H}^a_{dp}$ eingesetzten Kerneln $k_{\mathcal{F}_a,b}$ heißt "zulässig", wenn für jede Lipschitz-stetige Funktion $V \in C[0,1]^d$ der Zufallsoperator $\hat{H}^a_{dp}$ der Anforderung

$$||\hat{H}^a_{dp}V - H^a_{dp}V'||_\infty \xrightarrow{P} 0 \text{ wenn } m \to \infty \tag{4.45}$$

genügt.

Lemma 3. *Die Verkleinerungsrate b ist zulässig, wenn gilt $b^{d+1}\sqrt{m} \to \infty$, mit d Anzahl der Dimensionen von S, und $b \to 0$, wenn $m \to 0$.*

Der auf Ergebnissen von Rust [143] basierende Beweis dieses Lemmas ist ein zentrales Ergebnis auf dem Weg zum Beweis der Konsistenz von KADP. Aufgrund seiner hohen Komplexität und der wegen verletzter Grundannahmen – später erläuterten – geringen inhaltlichen Bedeutung für das DFQ-Verfahren sei an dieser Stelle auf die ausführliche Beweisführung in [113, Anhang A.2, Seite 173–175] und auf [143] verwiesen. Aus gleichem Grund sei auch lediglich auf die Berechnung einer optimalen Verkleinerungsrate in Theorem 3 und die Diskussion des Bias-Varianz-Dilemmas bei der Auswahl der passenden Rate in [113] verwiesen.

Lemma 4. *Unter jeder zulässigen Verkleinerungsrate $b(m)$ und jeder Lipschitz-stetigen Funktion $V \in C[0,1]^d$ gilt*

$$||\hat{H}V - HV||_\infty \xrightarrow{P} 0 \text{ wenn } m \to \infty. \tag{4.46}$$

Beweis. O.B.d.A. sei $S = [0,1]^d$. Dann gilt aufgrund der Endlichkeit von A, unter Verwendung der Korollar 3 nach [113, Anhang 2]:

$$||\hat{H}V - HV||_\infty = \sup_{s \in S} \left(\max_{a \in A} \hat{H}^a_{dp} V(s) - \max_{a \in A} H^a_{dp} V(s) \right) \tag{4.47}$$

$$\leq \sup_{s \in S} \max_{a \in A} \left(\hat{H}^a_{dp} V(s) - H^a_{dp} V(s) \right) \tag{4.48}$$

$$= \max_{a \in A} ||\hat{H}^a_{dp} V(s) - H^a_{dp} V(s)||_\infty \ . \tag{4.49}$$

Aus der Zulässigkeit von $b(m)$ entsprechend Definition 8 folgt dann direkt die zu beweisende stochastische Konvergenz gegen 0. □

Definition 8 sagt in Verbindung mit Lemma 3 nichts weniger aus, als dass es eine (praktische) Möglichkeit gibt, die "Kernelgröße" – gesteuert über die Bandbreite $b(m)$ – in solcher Weise zu verringern, dass der durch die Approximation eingeführte Fehler bei einmaliger Anwendung des Zufallsoperators $\hat{H}^a_{dp}$ gegenüber der Anwendung des exakten DP-Operators H^a_{dp} für größer werdende Stichproben gegen 0 verringert werden kann. Gleiches gilt laut Lemma 4 für die Schätzung $\hat{H}$ des DP-Operators H.

Für die Konsistenz gilt es noch zu beweisen, dass nicht nur der Fehler durch einmalige Anwendung des Zufallsoperators auf eine Ausgangsfunktion V mit m kleiner wird, sondern dass auch der in Satz 4 bestimmte Fixpunkt $\bar{V}$ der wiederholten Anwendungen des Zufallsoperators (stochastisch) gegen den Fixpunkt V^* des exakten Operators konvergiert, wenn die Größe m der Stichprobe gegen unendlich geht.

Satz 5 (Stochastische Konsistenz von KADP)**.** *Sei $b(m)$ eine zulässige Verkleinerungsrate der Bandbreite b des im Zufallsoperator $\hat{H}^a_{dp}$ eingesetzten Kernels, wobei m die Anzahl der Übergänge in der Stichprobe $\mathcal{F}$ bezeichne. Sei $\bar{V}$ weiterhin der Fixpunkt der approximativen Bellman-Gleichung $\hat{V}^{i+1} = H_{max} \hat{H}^a_{dp} \hat{V}^i$ des diskontierten MDPs mit $\gamma < 1$. Dann gilt unter den genannten Voraussetzungen*

$$||\bar{V} - V^*||_\infty \xrightarrow{P} 0 \text{ wenn } m \to \infty.$$

Beweis. Zum Beweis der stochastischen Konvergenz zeigen Ormoneit und Sen, dass die Wahrscheinlichkeit $P(||\bar{V} - V^*||_\infty > \epsilon)$ (P1) durch die Wahrscheinlichkeit $P(||\hat{H}V - HV||_\infty > (1-\gamma)\epsilon)$ (P2) beschränkt ist. P1 ist die Wahrscheinlichkeit, dass der Abstand des Fixpunktes $\bar{V}$ von der optimalen Wertfunktion V^* (in Max-Norm) weiter als die vorgegebene Schranke ϵ entfernt ist. Die Wahrscheinlichkeit P2 gibt an, wie wahrscheinlich es ist, dass der durch die einmalige Anwendung des Zufallsapproximators $\hat{H}$ gegenüber

H entstehende Fehler $(1-\gamma)\epsilon$ übersteigt. Dass P2 Schranke für P1 ist, folgt aus:

$$||\bar{V} - V^*|| \leq ||\bar{V} - \hat{H}V^*|| + ||\hat{H}V^* - V^*|| \tag{4.50}$$
$$= ||\bar{V} - \hat{H}V^*|| + ||\hat{H}V^* - HV^*|| \tag{4.51}$$
$$\leq \gamma||\bar{V} - V^*|| + ||\hat{H}V^* - HV^*|| \tag{4.52}$$
$$\equiv (1-\gamma)||\bar{V} - V^*|| \leq ||\hat{H}V^* - HV^*|| \ , \tag{4.53}$$

wobei $||\cdot||$ die Max-Norm bezeichne. Nach der Anwendung der Dreiecksungleichung in (4.50) folgt (4.51) direkt aus V^* per Definition Fixpunkt von $V^* = HV^*$. Die Abschätzung in (4.52) ergibt sich aus $||\bar{V} - \hat{H}V^*|| = ||\hat{H}\bar{V} - \hat{H}V^*||$ wegen $\bar{V} = \hat{H}\bar{V}$ und der Kontraktionseigenschaft von $\hat{H}$ laut Satz 4.

Im Falle stochastischer Konvergenz muss P1 für jedes $\epsilon > 0$ gegen 0 gehen, wenn $m \to \infty$. Da P2 für jede vorgegebene Schranke $(1-\alpha)\epsilon > 0$ gegen 0 geht, wenn $m \to \infty$ (Beweis der stochastischen Konvergenz gegen 0 in Lemma 4), nimmt damit auch die Wahrscheinlichkeit $P1$ wie gefordert für $m \to \infty$ ab (P2 ist obere Schranke auf P1). □

Ormoneit und Sen können bei der Auflösung von Gleichung (4.50) einen entscheidenden Schritt weiter als Gordon gehen, da sie mit den Lemmata 3 und 4 einen Weg gefunden haben, sowohl den durch die Mittelung als auch den durch die Verwendung einer Stichprobe eingeführten Fehler beliebig zu verkleinern. Dementsprechend betrachten sie nicht nur den durch die Mittelung eingeführten Fehler sondern lösen den Term $||\hat{H}V^* - V^*||$, in die andere Richtung als Gordon, nach $||\hat{H}V^* - HV^*||$ auf. Letztendlich führt dieser Weg mit der gegen 0 konvergierenden Abweichung zu einer wesentlichen stärkeren Aussage als die bisher zur Verfügung stehende obere Schranke bei FVI. Gerade für Algorithmen wie DFQ mit episodischer Exploration ist diese Konsistenzaussage besonders interessant.

Als abschließende Beobachtung sei noch angeführt, dass diese Konsistenzaussagen auch auf den von Gordon behandelten modellbasierten Fall übertragbar sind. Einerseits könnten alle Algorithmen und Beweise (aufwändig) mit exaktem DP-Operator anstelle der Schätzung $\hat{H}$ nachvollzogen werden. Andererseits ist es auch möglich, jeweils m Übergänge (perfekt verteilt!) von den bei Gordon bekannten Modellen zu ziehen und dann mit KADP weiterzurechnen.

4.1.5 Zusammenfassung

In diesem Abschnitt wurden die bestehenden Resultate zum modellbasierten und modellfreien batch RL wiedergegeben und in Zusammenhang gesetzt. Insbesondere die sich aus den Annahmen und Beweisen ergebenden Beschränkungen und Implikationen für praktische Anwendungen wurden ausführlich diskutiert. Die hinsichtlich der nachfolgenden Diskussion von DFQ und späterer Anwendungen zentralen Ergebnisse lassen sich wie folgt festhalten:

1. In KADP-Verfahren können aus den Transitionen $\mathcal{F}$ wahlweise Zustandswertfunktionen $V : S \mapsto \mathbb{R}$, Q-Funktionen $Q : S \times A \mapsto \mathbb{R}$ oder Aktionswertfunktionen

$Q_a : S \mapsto \mathbb{R}$ approximiert werden. Ein Wechsel von einer zur anderen Repräsentation ist zudem leicht über Korollar 3 und Gleichung (4.33) möglich. Die Approximation der Wertfunktionen folgt im KADP Verfahren separat für jede Aktion. Dazu wird die Stichprobe $\mathcal{F} = \{(s_t, a_t, r_{t+1}, s_{t+1}) |\ t = 1, \ldots, p\}$ entsprechend Definition 5 in Teilmengen $\mathcal{F}_a$ mit

$$\bigcup_{a \in A} \mathcal{F}_a = \mathcal{F} \quad \text{und} \quad \bigcap_{a \in A} \mathcal{F}_a = \emptyset$$

für die einzelnen Aktionen $a \in A$ aufgeteilt.

2. Es wurde eine enge Verwandtschaft zwischen Gordons und Ormoneit und Sens Untersuchungen bis hin zu den Beweiswegen festgestellt. Zentrale Idee von Ormoneit und Sen ist es, bei der Kostenschätzung über benachbarte Übergänge zu mitteln. Als Knackpunkt für den Nachweis der Konsistenz wurde die stochastische Aussage in Lemma 3 identifiziert. Die Aussagen zur Konsistenz gegeben Aufschluss darüber, wie das Ergebnis potenziell durch weitere Exploration verbessert werden kann. Mit der Vergrößerung der Stichprobe muss eine Absenkung des "Einzugsgebiet" bei der Mittelung erfolgen, um im Erwartungswert zu besseren Ergebnissen zu gelangen.

3. Für den diskontierten Fall lassen sich klare Konvergenzaussagen für das batch RL treffen, die auf der nicht-expansiven Eigenschaft sogenannter Averager basieren. Andere Verfahren, bei denen einzelne Stützstellen oder Datenpunkte negativ oder mit einem Gewicht größer als Eins gewichtet werden, führen unter Umständen zu Divergenz (siehe Korollar 1, Satz 2 und Definition 1 des Averagers, bzw. Definition 5 der Kernel und auch [97, 113]).

4. Kürzester Pfad Probleme mit $\gamma = 1$ führen bei ungeeigneter ("inkompatibler") Auswahl der Gewichte zu Problemen im Konvergenzverhalten und sind deutlich schwerer zu analysieren. Tatsächlich ist eine Aussage, ob eine Kombination von Gewichten und MDP zu einem stabilen Verhalten führt, nicht ohne grundsätzliche Kenntnis des Übergangsmodells T möglich. Umgangen werden kann dieses Problem in der Praxis durch eine geringe Absenkung des Diskontierungsfaktors auf einen geeigneten Wert $\gamma < 1$, der die optimale Strategie nicht (wesentlich) beeinflusst. Die "non-zero-weights"-Heuristik ist eine alternative Methode, Divergenz über die sichere Herstellung von Kompatibilität zu verhindern.

5. Das modellfreie KADP-Verfahren ist unter bestimmten Bedingungen stochastisch konsistent. Konkret bedeutet dies, dass die berechnete Wertfunktion in der Erwartung näher an der optimalen Wertfunktion liegt, wenn die Anzahl der Beobachtungen in der Stichprobe erhöht und die Kernelbandbreite geeignet verringert wird. Die Idee, den Einzugsbereich des Kernels stetig zu verkleinern, um mit der Schätzung näher an die optimale Wertfunktion zu gelangen, geht auf die kleiner werdenden Zellen einer Partitionierung [26] zurück und wurde auch schon von Gordon aufgegriffen (siehe Seite 59), bevor sie nun von Ormoneit und Sen für das

stichprobenbasierte Lernen perfektioniert wurde. Diese für KDADP hergeleiteten Resultate lassen sich prinzipiell auch auf den modellbasierten Fall übertragen – Erhöhung der Datenmenge bei gleichzeitiger Verringerung der Kernelbandbreite entspricht einer Erhöhung der Anzahl der Stützstellen.

6. Während der Beweis der Konvergenz kaum Bedingungen an die zu approximierende, unbekannte Wertfunktion und die Gewichtung und Verteilung der Stützstellen (FVI, Gordon) bzw. gezogenen Übergänge (KADP, Ormoneit et al.) stellt, basiert die Konsistenz auf härteren Annahmen und schränkt auch die verwendbaren Averager bzw. Kernelfunktionen [113, Anhang A.1, Assumption 4] ein. Insbesondere die geforderte Stetigkeit der Rewardsstruktur über alle Aktionen [113, Anhang A.1, Assumption 1] stellt ein ernstes Problem dar und ist in der Praxis bei wenigen System gegeben. Weder die Grid-World, noch das Mountain-Car oder das inverse Pendel erfüllen diese Anforderung. Dies bedeutet, dass es bei diesen Problemen auch durch eine Erhöhung der Stichprobengröße nicht möglich ist, den erwarteten Approximationsfehler (in Max-Norm) unter jede beliebige Schranke abzusenken. Aufgrund der Sprünge in der Wertfunktion bleibt immer irgendwo in der Nähe der Diskontinuität ein Restfehler, der sich nicht im Sinne der Lipschitz-Stetigkeit beschränken lässt. In der Praxis verhalten sich die genannten Systeme erfahrungsgemäß dennoch gutmütig. Man muss sich allerdings bewusst sein, dass trotz einer beliebig großen Datenmenge und guter Testergebnisse in der Nähe von Diskontinuitäten immer Restfehler bestehen bleiben.

4.2 Fitted Q-Iterations ist Variante des KADP-Verfahrens

Interessant ist die Feststellung, dass die Ergebnisse von Ormoneit et al. auch durchweg für den in DFQ eingesetzten FQI-Algorithmus von Ernst [39] gelten. Tatsächlich ist FQI nämlich nur eine Variante des kernelbasierten approximativen dynamischen Programmierens, die vollständig unter die Definitionen 4, 5 und 7 fällt. Dies wird deutlicher, wenn man die Aktualisierungsvorschrift des KADP-Verfahrens in Gleichung (4.37) und (4.38) mit der Aktualisierungsregel des FQI-Verfahrens (2.18) unter Einbeziehung der Gleichung (2.20) vergleicht. Die sichtbaren Unterschiede entstehen durch eine andere Auswahl der zwischengespeicherten Kostenschätzungen und durch eine veränderte Anordnung des DP-Schritts und des Mittelungsschritts. So wird in Gleichung (2.18) zwar der gleiche DP-Schritt wie in (4.37) vollzogen, dabei wird aber auf eine approximierte Q-Funktion $\hat{Q}^i$ an Stelle einer Zustandswertfunktion $\hat{V}^i$ zurückgegriffen.

Der FQI-Algorithmus legt mit seiner Formulierung einen deutlichen Schwerpunkt auf die Herausstellung der Speicherung der errechneten Schätzungen in einem (kernelbasierten) Funktionsapproximator, was ihn scheinbar näher an die Ideen Gordons rückt. In der Praxis bieten die Verwendung einer Q-Funktion und die explizite Speicherung ihrer Werte einen wichtigen Vorteil. Weil die aktuellen Q-Werte von FQI direkt in einem eigenständigen Funktionsapproximator (zum Beispiel in einem Gitter oder einem Baum) gespeichert werden, muss in der praktischen Anwendung nicht erst der komplizierte Berechnungsschritt entsprechend Gleichung (4.33) inklusive DP-Schritt und

Mittelung über alle Transitionen $(s, a, r, s') \in \mathcal{F}_a$ wie in KADP [113, Abschnitt 4, erster Absatz] vollzogen werden. Für die Theorie macht die Verwendung der Q-Funktion keinen Unterschied, denn durch Maximierung über die Aktionen ist es leicht möglich, eine Zustandswertfunktion aus einer Q-Funktion zu berechnen. Genau dieser Schritt wird im KADP Verfahren in Gleichung (4.38) vollzogen, könnte offensichtlich aber auch direkt in die Aktualisierung in Gleichung (4.37) hineingezogen werden.

Der andere sichtbare Unterschied besteht in dem Zeitpunkt der Durchführung der Mittelwertbildung über benachbarte Transitionen. In KADP wird die kernelbasierte Mittelung direkt in (4.37) durchgeführt, in FQI wird der im DP-Schritt für eine Transition errechnete Q-Zielwert zunächst für den Ausgangszustand s_t in $\bar{q}_t$ zwischengespeichert. Die Mittelung über benachbarte Transitionen – in FQI über deren zugehörige Werte $\bar{q}_t$ – wird dann bei der Abspeicherung in einem kernelbasierten Approximator (Schritt 3 von FQI) in Gleichung (2.20) vollzogen. Auch diese Änderung bei FQI führt formal zu einem zu (4.37) und (4.38) äquivalenten Algorithmus, was bei Einsetzen von (2.18) in (2.20) sofort deutlich wird:

$$f_a(s_0) = \sum_{(s,a;\bar{q})\in\mathcal{P}_a} k(s, s_0)\bar{q} \tag{4.54}$$

$$= \sum_{(s,a,r,s')\in\mathcal{F}_a} k(s, s_0)\left[r + \gamma \max_{a'\in A} \hat{Q}^i(s', a')\right] . \tag{4.55}$$

(4.54) ist der ursprüngliche Mittelungsschritt (2.20) im FQI-Verfahren, aus dem sich sofort (4.55) durch das Einsetzen des im FQI-Verfahren vollzogenen DP-Schrittes $\bar{q}^{i+1} \leftarrow r + \gamma \max_{a'\in A} \hat{Q}^i(s', a')$ (2.18) ergibt. Die Form (4.55) ist praktisch identisch zu der Gleichung, die sich durch Einsetzen von (4.38) in (2.18) für KADP ergibt:

$$\hat{Q}_a^{i+1}(s_0) = \sum_{(s,a,r,s')\in\mathcal{F}_a} k(s, s_0)[r + \gamma \hat{V}^i(s')] \tag{4.56}$$

$$= \sum_{(s,a,r,s')\in\mathcal{F}_a} k(s, s_0)[r + \gamma \max_{a\in A} \hat{Q}_a^i(s_0)] . \tag{4.57}$$

Die Umordnung der Operationen beim FQI-Verfahren legt lediglich ein stärkeres Gewicht auf eine Trennung des Funktionsapproximators vom dynamischen Programmieren, während bei Ormoneit und Sen aus Gründen der einfacheren Beweisführung beide Schritte in einer Gleichung zusammengefasst sind. Bestärkt wird durch die Darstellung in (2.18) und (2.20) zudem, dass in der Iteration von FQI Aktionswerte explizit zwischengespeichert werden, während KADP nach [113, Abschnitt 4, erster Absatz] vorzugsweise (kann, muss nicht) die Zustandswerte der Endpunkte der Transitionen explizit repräsentiert.

Auch wenn Ernst in [39, Abschnitt 3.6] die genannten Unterschiede seines Algorithmus zum KADP-Verfahren und die Nähe zu Gordons approximativer Wertiteration heraushebt, handelt es sich beim FQI-Verfahren für endliche Aktionsmengen A um eine spezielle Variante des KADP-Verfahrens. Die eingesetzten Aktualisierungsvorschriften und auch der kernelbasierte Funktionsapproximator werden durch die Definitionen 5

und 7 erfasst und führen unter formalen Gesichtspunkten letztendlich zu einem zu Definition 7 äquivalenten (bzw. identischen) Algorithmus. Diese Verwandtschaft wird auch bei Ernst deutlich, zum Beispiel in der Ableitung des Beweises zu Satz 4 in [39, Anhang B] aus Ormoneit und Sen [113, Theorem 1, Anhang A.2].

Mit FQI als einer Variante des KADP-Verfahrens sind Ormoneit und Sens Ergebnisse auf FQI anwendbar und damit auch für DFQ relevant. In der folgenden Analyse von DFQ wird daher weiter mit ihnen gearbeitet.

4.3 Situation in DFQ

Das DFQ-Verfahren unterscheidet sich von den bisher diskutierten herkömmlichen batch RL-Verfahren FVI, KADP und FQI darin, dass Strategien nicht direkt auf dem Zustandsraum S, sondern auf einem Merkmalsraum Z erlernt werden, der automatisch aus den im Anwendungsszenario statt der Zustände erhaltenen Observationen $o \in O$ abgeleitet wird. Diese ausführlich in Kapitel 3 beschriebene, veränderte Situation innerhalb von DFQ ist in Abbildung 4.2 schematisch dargestellt und lässt sich wie folgt zusammen fassen.

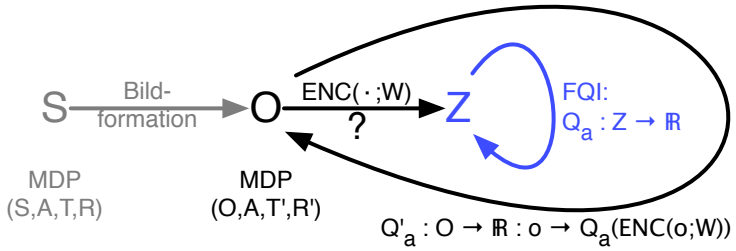


Abbildung 4.2: Grafische Darstellung der Zusammenhänge zwischen Zuständen, Observationen und Merkmalsvektoren mit der in DFQ approximierten Q-Funktion.

Aufgrund der in Abschnitt 3.2 geforderten Linkseindeutigkeit und Informationserhaltung der Bildformation verhält sich eine optimale Strategie auf dem Observationen-MDP auch optimal hinsichtlich des ursprünglichen Systemzustands-MDPs. Aus diesem Grund kann in der Analyse des DFQ-Algorithmus der in der Zeichnung leicht angegraute Bereich mit Systemzuständen und Bildformation außer Betracht bleiben.

Weil das Erlernen einer Strategie direkt auf den hoch-dimensionalen Observationen an praktischen Gesichtspunkten scheitert, werden innerhalb von DFQ die beobachteten Observationen $o \in O$ zunächst mittels eines Encoders $\mathrm{ENC}(\,\cdot\,; W)$ mit Gewichten W in einen Merkmalsraum Z abgebildet. Auf Basis dieses Merkmalsraums werden dann die Fitted-Q Iterations angewendet, um eine Q-Funktion Q bzw. Aktionswertfunktionen Q_a für jede Aktion $a \in A$, zu erlernen, die den zu einer Observation gehörenden Merkmalsvektor $z = \mathrm{ENC}(o, W)$ auf die zu erwartenden (optimalen) Kosten bei An-

wendung der Aktion a abbildet. Ein Ergebnis dieses Kapitels wird es sein, dass es zu der für den Merkmalsraum erlernten Q-Aktionswertfunktion Q_a eine korrespondierende Wertfuntktion $Q'_a : O \mapsto \mathbb{R}$ für den Observationsraum gibt, von der eine Strategie für das Observationen-MDP abgeleitet werden kann.

Konvergenz Wegen dieser gegenüber dem "Standard"-Vorgehen auf dem Observationen-MDP (O, A, T_O, R_O) veränderten Bedingungen ergeben sich gleich mehrere Problemfelder hinsichtlich Konvergenz und Konsistenz (Übersicht in Abbildung 4.3) von DFQ. So erscheint eine stabile Konvergenz des auf den Merkmalsraum angewendeten FQI-Algorithmus in der inneren Schleife von DFQ zunächst eher fraglich. Die Merkmalsvektoren sind im Allgemeinen keine markovsche Zustandsrepräsentation, da es durch die Abbildung auf den niedrigdimensionalen Raum zu einem generellen Verlust wichtiger Informationen und zu Zustandsverdeckungen kommen kann. Es lässt sich somit in der Regel kein Merkmals-MDP (Z, A, T_z, R_z), sondern nur ein POMDP auf Z formulieren. Die Voraussetzungen zur Anwendung von batch RL und der Ergebnisse von Gordon, Ernst und Ormoneit und Sen scheinen damit schon im Grundsatz verletzt.

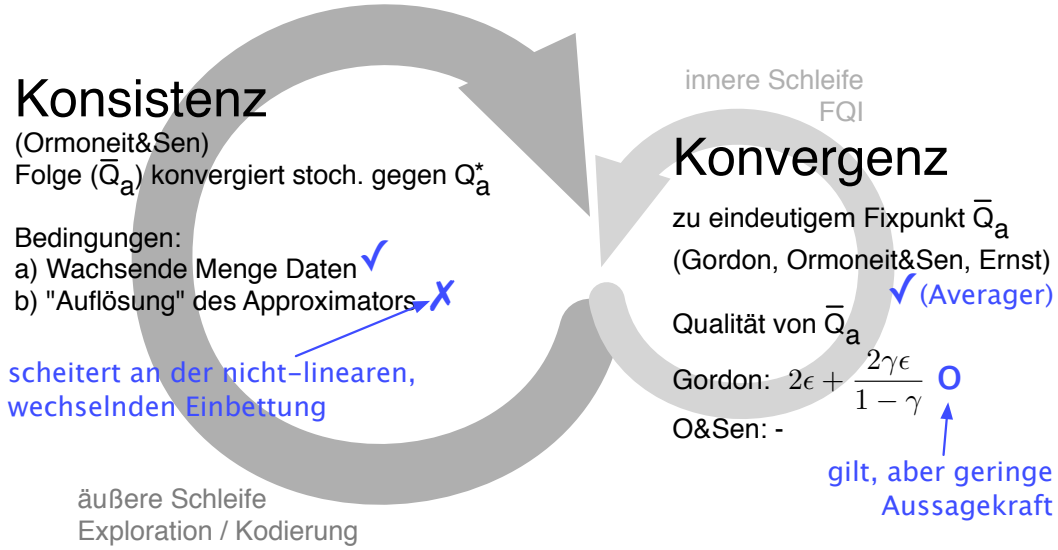


Abbildung 4.3: Grafische Darstellung der Relevanz der bestehenden theoretischen Ergebnisse zur Konvergenz und Konsistenz für den Ablauf von DFQ. Im Vorgriff auf die weiteren Untersuchungen wurden die Ergebnisse bezüglich DFQ in blau über die Grafik gelegt.

Durch die diskutierten theoretischen Ergebnisse zur Konvergenz gedeckt ist lediglich die Anwendung von FQI und KADP auf das Observationen-MDP. Soll ausgehend von dieser sicheren Basis argumentiert werden, muss der Encoder als Teil des eingesetzten Funktionsapproximators interpretiert werden: Die Kosten für die Observationen

$o \in O$ werden nicht durch eine Mittelung über benachbarte Observationen geschätzt, sondern über eine Mittelung über die Merkmalsvektoren, die durch den Encoder mittels einer nichtlinearen Abbildung erzeugt werden. Solchermaßen als Teil des Approximators aufgefasst, scheint die in der Mittelung vorgenommene, nichtlineare Abbildung von Observationen auf Merkmalsvektoren den Bedingungen an die Averager aus Definition 1 und dem Korrolar 1 zu widersprechen, die allgemeine, nichtlineare Funktionsapproximatoren aufgrund möglicher expansiver Abbildungen ausschließen. Diese Bedingungen finden sich ebenfalls in den Anforderungen (4.34) und (4.35) an die Gewichtung der Trainingsbeispiele im KADP-Verfahren wieder.

Tatsächlich verursacht diese "Vorschaltung" des Encoders vor einen Averager aber kein Problem hinsichtlich der Konvergenz nach Satz 2 und Satz 4. Der Grund ist, dass die vom Encoder realisierte Abbildung nicht auf die Zielwerte selbst (z.B. die Q-Werte $\bar{q}_t$) angewendet wird, sondern nur der Bestimmung der Gewichte dient, mit denen die einzelnen Zielwerte in die Mittelung eingehen. Diese Gewichte können in KADP-Verfahren wie bei FVI auf beliebige Art und Weise bestimmt werden, auch durch eine nichtlineare Abbildung und ohne Berücksichtigung der Nachbarschaftsbeziehungen zwischen den Zuständen des MDPs, solange sie nur den genannten Bedingungen (4.34) und (4.35) (nicht-negativ, Summation zu 1) genügen und sich während der Durchführung des approximativen dynamischen Programmierens – wie in der inneren Schleife von DFQ – nicht ändern. Zwar setzen Gordon und Ormoneit und Sen in ihren Arbeiten MDPs voraus [53, 113], die Beweise der Konvergenz auf diskontierten Problemstellungen (siehe Satz 2 und Satz 4) greifen aber gar nicht auf die Markov-Eigenschaft der Zustandsbeschreibung zurück, sondern erfordern lediglich die Diskontierung mit $\gamma < 1$ und die Beschränktheit der sofortigen Belohnungen r. Damit gelten sie auch ganz allgemein für POMDPs und insbesondere auch für die Anwendung von FQI auf den Merkmalsraum in DFQ.

Speziell für DFQ lässt sich darüber hinaus zeigen, dass eine Ausführung des FQI-Algorithmus mit einem Averager A auf dem Merkmalsraum direkt zu der Ausführung des FQI-Algorithmus mit einem bestimmten anderen Averager B auf dem Observationen-MDP korrespondiert und beide Vorgehensweisen zu identischen Ergebnissen führen. Die Gewichte des dazu notwendigen Averagers B hängen neben dem verwendeten Averager A auch vom in DFQ eingesetzten Encoder ab und lassen sich auf eindeutige Weise bestimmen. Über die Konstruktion einer korrespondierenden Kernelfunktion gelingt es, die nichtlineare Abbildung der Observationen in den Gewichten der Mittelung quasi verschwinden zu lassen, ohne die in KADP und FVI an die Gewichte gestellten Bedingungen zu verletzen.

Diese positiven Argumente zur Konvergenz von DFQ und zur Korrespondenz der gefundenen Lösungen auf dem Observationen-MDP und dem Merkmalsraum werden im folgenden Abschnitt 4.4 formal vollzogen. Die fehlende Markov-Eigenschaft der Merkmalsvektoren und die Verwendung des Encoders spielen dann aber doch eine Rolle bei der in Abschnitt 4.4.4 diskutierten Kompatibilität der eingesetzten Funktionsapproximatoren und Qualität der gefundenen Lösungen. Hier setzen alle Aussagen Gordons und Ormoneit und Sens zwingend ein MDP und die Existenz der nach Definition 2.3

definierten optimalen Wertfunktion voraus.

Konsistenz Anders als FQI ist DFQ kein reiner (offline) batch Algorithmus, sondern als semi-online Algorithmus mit Exploration ausgelegt. Mit jeder weiteren Iteration der äußeren Schleife wird der FQI-Algorithmus in der inneren Schleife auf die mit jeder Explorationsepisode an Zahl zunehmenden Transitionsdaten angewendet. Diesbezüglich sind insbesondere die Resultate von Ormoneit und Sen zur Konsistenz interessant. Sie geben erstmals eine über die intuitive Motivation hinausgehende formale Begründung für semi-online Varianten und zeigen einen möglichen Vorteil gegenüber reinen batch Algorithmen auf, während sich bei Ernst [39] – wohl aufgrund der Verwendung eines ungeeigneten Approximators – in der Beschreibung des FQI-Algorithmus gar keine Aussagen zu einer wiederholten Anwendung bzw. zu seiner Konsistenz finden. In der Anwendung von Ormoneit und Sens Ergebnissen zur Konsistenz auf DFQ in Abschnitt 4.5 ergeben sich allerdings Probleme mit den nichtlinearen Encodern, aufgrund deren Einsatz wichtige Annahmen in der Beweisführung verletzt sind. In Verbindung mit den Generationswechseln ergibt sich eine weitere, formal momentan nicht zu schließende Lücke.

4.4 Zur Konvergenz in der inneren Schleife

Um die Analyse des DFQ-Algorithmus zu erleichtern, wird zunächst ein Zusammenhang zwischen den auf einem Merkmalsraum Z berechneten Wertfunktionen $V : Z \mapsto \mathbb{R}$ und Wertfunktionen $V' : O \mapsto \mathbb{R}$ auf dem Observationen-MDP hergestellt. Obwohl der DFQ-Algorithmus den FQI Algorithmus auf die Übergänge $\mathcal{F}_Z$ im Merkmalsraum Z anwendet und als Ergebnis eine Q-Wertfunktion und Strategie im Merkmalsraum Z zurückliefert, wird es über die hergestellte Korrespondenz dann möglich, eine Analyse auf den Observationen vorzunehmen. Dieses Vorgehen hat den Vorteil, dass die Markov-Eigenschaft der Observationen bekannt ist und sich im Observationsraum anders als auf den Merkmalsvektoren ein MDP formulieren lässt. Für diesen Fall wurde die Konvergenz des KADP-Verfahrens bereits nachgewiesen. Von Interesse ist letztendlich eine möglichst gute (optimale) Strategie $\pi : O \mapsto A$ für die Auswahl von Aktionen auf Basis der erhaltenen Observationen.

4.4.1 Definition korrespondierender Wertfunktionen

Mit Hilfe folgender Definition können von einer beliebigen Q-Funktion $Q : Z \times A \mapsto \mathbb{R}$, Aktionswertfunktion $Q_a : Z \mapsto \mathbb{R}$ oder Zustandswertfunktion $V : Z \mapsto \mathbb{R}$ auf dem Merkmalsraum Z leicht entsprechende Wertfunktionen für den Observationsraum $\mathcal{O}$ abgeleitet werden – und das in eindeutiger Weise.

Definition 9. Zu jeder auf dem Merkmalsraum definierten Q-Funktion $Q : Z \times A \mapsto \mathbb{R}$ gibt es eine korrespondierende Q-Funktion $Q' : O \times A \mapsto \mathbb{R}$, die für jede Aktion $a \in A$ und zu jeder Beobachtung o dieselben Kosten wie sie Q für die gleiche Aktion und den zu o gehörigen Merkmalsvektor $z = \mathrm{ENC}(o; W)$ liefert. Diese korrespondierende

Funktion kann leicht durch eine Verknüpfung der Q-Funktion Q mit dem Encoder ENC berechnet werden:

$$\forall(o,a) \quad Q'(o,a) = Q(\mathrm{ENC}(o;W),a) \tag{4.58}$$

Analog ist die zur auf dem Merkmalsraum definierten Aktionswertfunktion $Q_a : Z \mapsto \mathbb{R}$ korrespondierende Aktionswertfunktion $Q'_a : O \mapsto \mathbb{R}$ gegeben durch

$$\forall o \quad Q'_a(o) = Q_a(\mathrm{ENC}(o;W)) \tag{4.59}$$

und für die korrespondierenden Zustandswertfunktionen $V : Z \mapsto \mathbb{R}$ und $V' : O \mapsto \mathbb{R}$ gilt gleichfalls

$$\forall o \quad V'(o) = V(\mathrm{ENC}(o;W)). \tag{4.60}$$

4.4.2 Konstruktion eines korrespondierenden Zufallsoperators

Im nächsten Schritt soll nun ein Zufallsoperator konstruiert werden, der es erlaubt, einen einzelnen Schritt des KADP-Verfahrens auf den Observationen durchzuführen, wobei der ausgeführte Schritt in bestimmter Hinsicht äquivalent zum Vollziehen eines vorgegebenen KADP-Schrittes im Merkmalsraum ist. Für diesen Zweck wird zunächst eine zu Definition 9 ähnliche Formulierung für korrespondierende Kernelfunktionen erarbeitet.

Lemma 5. *Sei $\mathcal{F}_{Z,a} = \{(z_t, a_t, r_{t+1}, z_{t+1}) \in \mathcal{F}_Z |\ t = 1,\ldots,p \ \wedge a_t = a\}$ die Teilmenge von $\mathcal{F}_Z$, die alle Übergänge aus $\mathcal{F}_Z$ enthält, in denen Aktion a gewählt wurde. Sei $k(z,z')$ eine auf $\mathcal{F}_{Z,a}$ definierte Gewichtsfunktion entsprechend den Bedingungen (4.34) und (4.35). Sei $\mathcal{F}_{O,a}$ die Menge der in $\mathcal{F}_{Z,a}$ enthaltenen Transitionen im Beobachtungsraum, also $z_i = ENC(o_i;W)$, $z'_i = ENC(o'_i;W)$ für alle $(z_i,a_i,r_i,z'_i) \in \mathcal{F}_{Z,a}$. Der zu $k(z,z')$ korrespondierende Kernel $k'(o,o')$ auf den Observationen ist definiert als*

$$k'(o,o') := k(ENC(o;W), ENC(o';W)) \ .$$

Der so definierte Kernel k' erfüllt die Bedingungen (4.34) und (4.35) auf $\mathcal{F}_{O,a}$.

Beweis. Dass $k'(o,o') \geq 0$ (Bedingung (4.34)) für alle $o,o' \in O$ folgt daraus, dass mit $z = \mathrm{ENC}(o;W)$, $z' = \mathrm{ENC}(o';W)$ gilt:

$$\begin{aligned} k'(o,o') &= k(\mathrm{ENC}(o;W), \mathrm{ENC}(o';W)) \\ &= k(z,z') \\ &\geq 0 \ , \end{aligned}$$

denn per Definition $k(z,z') \geq 0$ für alle $z,z' \in Z$. Für die Summe der Gewichte (Bedingung (4.35)) gilt:

$$\begin{aligned} \sum_{(o,a,r,o') \in \mathcal{F}_{O,a}} k'(o,o') &= \sum_{(o,a,r,o') \in \mathcal{F}_{O,a}} k(\mathrm{ENC}(o;W), \mathrm{ENC}(o';W)) \\ &= \sum_{(z,a,r,z') \in \mathcal{F}_{Z,a}} k(z,z') = 1 \ , \end{aligned} \tag{4.61}$$

denn $k(z,z')$ efüllt per Voraussetzung Bedingung (4.35) auf $\mathcal{F}_{Z,a}$. □

Solch eine Gewichtsfunktion für die Übergänge $\mathcal{F}_{O,a}$, die mit Hilfe einer vorgegebenen, geeigneten Gewichtsfunktion $k(z, z')$ auf $\mathcal{F}_{Z,a}$ und einem Encoder ENC($\cdot$; W) konstruiert wurde, darf deshalb in einem Zufallsoperator für den Observationsraum verwendet werden.

Lemma 6. *Sei $\mathcal{F}_{O,a}$ eine Menge von Transitionen im Observationsraum und $\mathcal{F}_{Z,a}$ die Menge zugehöriger Transitionen im Merkmalsraum mit $z_i = ENC(o_i; W)$ und $z'_i = ENC(o'_i; W)$. Zu jedem Zufallsoperator $\hat{H}^a_{dp}$ auf $\mathcal{F}_{Z,a}$ mit*

$$\hat{H}^a_{dp} V(z_0) = \sum_{(z,a,r,z') \in \mathcal{F}_{Z,a}} k(z, z_0)[r + \gamma V(z')]$$

gibt es einen korrespondierenden Zufallsoperator

$$\widetilde{H}^a_{dp} V(o_0) = \sum_{(o,a,r,o') \in \mathcal{F}_{O,a}} k'(o, o_0)[r + \gamma V'(o')]$$

auf $\mathcal{F}_{O,a}$, wobei gilt $\widetilde{H}^a_{dp} V'(o_0) = \hat{H}^a_{dp} V(z_0)$ für alle $o_0 \in O$ und $z_0 = ENC(o_0; W)$. Dabei ist $V'(o)$ die zu $V(z)$ korrespondierende Wertfunktion nach Definition 9.

Beweis. Sei $k'(o, o') := k(ENC(o; W), ENC(o'; W))$ entsprechend Lemma 5 zu $k(z, z')$ korrespondierender Kernel. Sei $V'(o') = V(ENC(o'; W))$ zu $V(z)$ korrespondierende Wertfunktion entsprechend 4.60 in Definition 9. Dann folgt für alle $o_0 \in O$:

$$\begin{aligned}
\widetilde{H}^a_{dp} V'(o_0) &= \sum_{(o,a,r,o') \in \mathcal{F}_{O,a}} k'(o, o_0)[r + \gamma V'(o')] && (4.62)\\
&= \sum_{(o,a,r,o') \in \mathcal{F}_{O,a}} k(ENC(o; W), ENC(o_0; W))[r + \gamma V(ENC(o'; W))] && (4.63)\\
&= \sum_{(z,a,r,z') \in \mathcal{F}_{Z,a}} k(z, ENC(o_0; W))[r + \gamma V(z')] && (4.64)\\
&= \hat{H}^a_{dp} V(z_0) \ , && (4.65)
\end{aligned}$$

wobei $z_0 = ENC(o_0; W)$.[1] □

Mit Hilfe dieser Ergebnisse ist es in der Folge nun möglich, den auf $\mathcal{F}_{Z,a}$ angewendeten KADP-Algorithmus auch auf dem Observationen-MDP zu untersuchen. Wird mit Hilfe des auf $\mathcal{F}_{Z,a}$ definierten Zufallsoperators $\hat{H}^a_{dp}$ aus der Funktion $V^i : Z \mapsto \mathbb{R}$ eine Aktionswertfunktion $\hat{Q}^{i+1}_a : Z \mapsto \mathbb{R}$ mit $\hat{Q}_a = \hat{H}^{DP}_a V^i$ nach Vollzug eines approximativen DP-Schrittes berechnet, gibt es nach Definition 9 sowohl zur Ausgangsfunktion V^i eine korrespondierende Wertfunktion $V'^i : O \mapsto \mathbb{R}$ als auch eine zum Resultat $\hat{Q}^{i+1}_a$ korrespondierende Aktionswertfuntion $\hat{Q}'^{i+1}_a : O \mapsto \mathbb{R}$ auf den Observationen. In Hinblick auf DFQ fällt zudem auf, dass es genau die korrespondierende Funktion $\hat{Q}'^{i+1}_a$ ist,

[1] Auch die Eindeutigkeit des korrespondierenden Zufallsoperators $\widetilde{H}^a_{dp}$ für das Observationen-MDP lässt sich beweisen, spielt aber für die Argumentation an dieser Stelle keine Rolle und wurde deshalb in Anhang A verlegt.

an der man für die Ableitung einer Strategie eigentlich interessiert ist, denn man kann in Schritt F von DFQ leicht eine Strategie von dieser korrespondierenden Wertfunktion nach

$$\pi^{i+1}(o) = \arg\max_{a \in A} \hat{Q}_a^{i+1}(z, a) = \arg\max_{a \in A} \hat{Q}_a'^{i+1}(o, a)$$

ableiten (vgl. auch Ableitung der Strategie in DFQ auf Seite 47), wobei $z = \mathrm{ENC}(o; W)$. Entscheidend ist nun die Feststellung, dass man die Q-Funktion Q'^{i+1} auch "direkt" im Observationsraum durch die Anwendung des zu $\hat{H}_{dp}^a$ korrespondierenden Zufallsoperators $\widetilde{H}_{dp}^a$ für $\mathcal{F}_{O,a}$ auf $\hat{V}'^i$ berechnen kann, denn nach Lemma 6: $Q'^{i+1} = \widetilde{H}_{dp}^a \hat{V}'^i$.

Statt zunächst alle Observationen mit dem Encoder in den Merkmalsraum zu übersetzen, um dann den Operator $\hat{H}_{dp}^a$ auf die Merkmalsvektoren anzuwenden, könnte also theoretisch auch der korrespondierende Operator $\widetilde{H}_{dp}^a$ direkt auf die Observationen angewendet werden, ohne die Merkmalsvektoren dauerhaft zwischenspeichern bzw. die Menge $\mathcal{F}_{Z,a}$ jemals explizit berechnen zu müssen. Mit der Konstruktion des korrespondierenden Operators wurde die durch den Encoder realisierte nichtlineare Abbildung der Observationen auf ihre Merkmalsvektoren quasi in die Gewichte der kernelbasierten Mittelung hineingezogen.

4.4.3 Herleitung der Konvergenz zu einem eindeutigen Fixpunkt

Der verbleibende Beweis der Konvergenz von KADP innerhalb von DFQ ist mit Hilfe der Vorarbeiten für diskontierte MDPs nun fast trivial, da der wesentliche Schritt bereits im vorangegangenen Abschnitt mit der Konstruktion der korrespondierenden Kernel und Zufallsoperatoren vollzogen wurde.

Satz 6. *Die Anwendung des KAPD-Algorithmus mit Kernel k und Trainingsmenge $\mathcal{F}_Z$ konvergiert für einen Diskontierungsfaktor $\gamma < 1$ zu einem eindeutigen Fixpunkt $\bar{V} : Z \mapsto \mathbb{R}$.*

Beweis. Intuitiv lässt sich die Konvergenz von KADP in der inneren Schleife von DFQ gegen einen eindeutigen Fixpunkt schlicht dadurch begründen, dass die Berechnung eines Schrittes der Anwendung des KADP-Ansatzes auf das Observationen-MDP mit einem korrespondierenden Zufallsoperator entspricht und dass dieser äquivalente Algorithmus auf dem Observationen-MDP nach Satz 4 sicher konvergiert. Der auf dem Observationen-MDP berechnete Fixpunkt $\bar{V}'$ wäre die zu $\bar{V} : Z \mapsto \mathbb{R}$ korrespondierende Funktion, die nach Lemma 6 durch eine gleiche Anzahl von Iterationen des KADP-Updates auf $\mathcal{F}_Z$ berechnet werden würde.

Formal ergibt sich analog zum Beweis von Satz 4 mit $z = \mathrm{ENC}(o; W)$ für alle $z \in Z$ (Z ist der vom Encoder $\mathrm{ENC}(\,\cdot\,; W)$ "aufgespannte" Merkmalsraum bzw. seine

Bildmenge):

$$||\hat{H}V_1 - \hat{H}V_2||_\infty = \sup_{z\in Z} \left| \max_{a\in A} \hat{H}^a_{dp} V_1(z) - \max_{a\in A} \hat{H}^a_{dp} V_2(z) \right| \tag{4.66}$$

$$= \sup_{o\in O} \left| \max_{a\in A} \widetilde{H}^a_{dp} V'_1(o) - \max_{a\in A} \widetilde{H}^a_{dp} V'_2(o) \right| \tag{4.67}$$

$$\leq \gamma ||V'_1 - V'_2||_\infty \tag{4.68}$$

$$= \gamma ||V_1 - V_2||_\infty \ , \tag{4.69}$$

wobei hier ausnahmsweise $V'_1 : O \mapsto \mathbb{R}$ und $V'_1 : O \mapsto \mathbb{R}$ die zu $V_1 : Z \mapsto \mathbb{R}$ respektive $V_2 : Z \mapsto \mathbb{R}$ korrespondierenden Funktionen nach Definition 9 bezeichnen. Dabei folgt (4.67) aus Lemma 6 und (4.68) folgt aus Satz 4, genauer aus Gleichungen (4.39-4.44). Konvergenz zu einem eindeutigen Fixpunkt $\bar{V} : Z \mapsto \mathbb{R}$ bzw. $\bar{V}' : O \mapsto \mathbb{R}$ ergibt sich direkt aus dem Banachschen Fixpunktsatz und (4.69). □

Durch die nichtlineare Einbettung tritt hinsichlich der Konvergenz bei $\gamma < 1$ demnach kein Problem auf. Ein auf den Merkmalsraum angewendetes KADP-Verfahren A entspricht einer Anwendung des KADP-Verfahrens B auf die Observationen, wobei sich A und B nur in den eingesetzten Kerneln unterscheiden. Beide Verfahren konvergieren zu einem eindeutigen Fixpunkt, wobei der Fixpunkt von B die zum Fixpunkt von A korrespondierende Funktion ist. Das ist ein hervorragendes Ergebnis für DFQ, denn es ist sichergestellt, dass die innere Schleife in jedem Fall stabil und zu einem eindeutigen Ergebnis konvergiert, ganz unabhängig davon, wie gut der in der äußeren Schleife berechnete Encoder gerade ist und von welcher initialen Wertfunktion aus gestartet wurde.

4.4.4 Ergebnisse für nicht diskontierte MDPs und die Fehlerabschätzung

Während die nichtlineare Einbettung hinsichtlich der Stabilität in diskontierten MDPs offenbar keine Rolle spielt, ist das Verstecken der Einbettung in den Gewichten der Approximation an anderer Stelle nicht ohne Kosten. So ist der Einzugsbereich des Kernels k' auf den Observationen auch von der durch den Encoder realisierten Abbildung $\text{ENC} : O \mapsto_W Z$ abhängig. Auch wenn für den Merkmalsraum z.B. ein Kernel k in Form einer Gaußglocke nach Definition 6 vorgegeben wurde, kann – wohlgemerkt "kann", nicht "muss" – es aufgrund der nicht näher spezifizierten Eigenschaften und der Nichtlinearität der Abbildung dazu kommen, dass der Einzugsbereich des korrespondierenden Kernels k' in ungünstigen Fällen im Observationsraum quasi zerrissen wird. Ein solches Beispiel ist für eine konkrete Abbildung von dreidimensionalen Observationen auf zweidimensionale Merkmalsvektoren in Abbildung 4.4 zu sehen. Obwohl der gewählte Kernel $k(z, z_o)$ im Merkmalsraum Z für einen Merkmalsvektor $z_0 = \text{ENC}(o_0; W)$ (oranger Punkt) nur Datenpunkte (schwarze Punkte) innerhalb eines engen Radius berücksichtigt, liegen die vom korrespondierenden Kernel k' berücksichtigten Observationen (euklidisch) nicht

in der Nachbarschaft von o_0. Vielmehr liegen die vom Kernel k' berücksichtigten Datenpunkte – die schwarz markierten Urbilder der schwarzen markierten Punkte in Z – in diesem speziellen Fall kreuz und quer im Observationsraum verteilt. Gleichzeitig kann durch eine ungünstige Abbildung der Observationen auf die Merkmalsvektoren die Markov-Eigenschaft der Observationen verloren gehen und es können Verdeckungen von Zuständen auftreten.

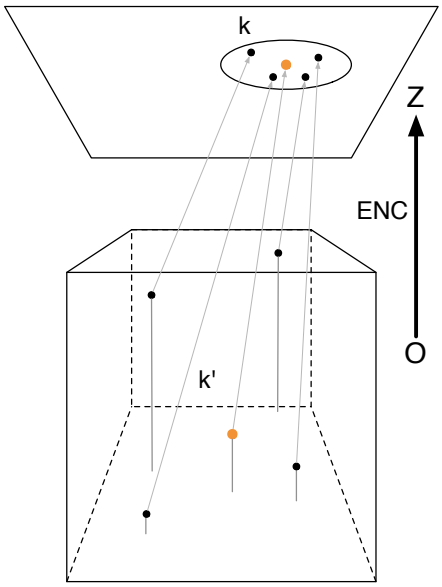


Abbildung 4.4: Grafische Darstellung der von einem Kernel k in einem zweidimensionalen Merkmalsraum Z erfassten Merkmalsvektoren und ihrer Urbilder im dreidimensionalen Observationsraum O.

Diese möglichen Eigenschaften der Abbildung haben unangenehme Auswirkungen auf die Analyse von nicht diskontierten Observationen-MDPs mit kürzester Pfad Modellierung und $\gamma = 1$. Kommt es zum Beispiel unter der Abbildung zu Verdeckungen von wichtigen Zuständen des MDPs (absorbierende Terminalzustände, Flaschenhälse), lassen sich im Merkmalsraum ganz unabhängig vom dort eingesetzten Approximator keine zielführenden Strategien erlernen und entsprechend auch keine kompatiblen Kernel finden. Diese Situation kann so interpretiert werden, dass der nichtlineare Anteil (ENC) von k' durch die Einbettung bereits in einer solchen Weise festgelegt wird, dass es keinen Kernel k auf den Merkmalen gibt, der $k'(o, o') = k(\mathrm{ENC}(o), \mathrm{ENC}(o'))$ zu einem kompatiblen Kernel nach Gordons Kriterien (siehe Diskussion nicht diskontierter MDPs auf Seite 57 und [53, Abschnitt 4]) macht. In der Praxis wird die allgemein schwierige Einschätzung der Kompatibilität des Kernels k' zum Observationen-MDP praktisch unmöglich. Die Auswahl eines kompatiblen Approximators vor dem Start des DFQ-Algorithmus wird in der Praxis zusätzlich dadurch erschwert, dass die genaue Einbettung zu diesem Zeitpunkt noch gar nicht bekannt ist, erst innerhalb von DFQ erlernt

und zudem laufend verändert wird.

Ein weiteres Problem entsteht bei der Beurteilung der Güte der resultierenden Wertfunktion. Prinzipiell gilt Gordons obere Schranke in Satz 3 auch, wenn FVI auf die Merkmalsvektoren angewendet wird. Der Intuition nach gibt sie auch einen groben Anhaltspunkt dafür, was mit KADP möglich wäre, wenn die Übergangswahrscheinlichkeiten durch den Zufallsoperator perfekt geschätzt würden; also bei idealer Datenlage und kleiner Bandbreite des Kernels (siehe Diskussion des Q-Lernens aus Sicht von FVI auf Seite 60 und [113, Anhang A.2]). Nach Gordon hängt die Schranke über die Differenz $\epsilon = ||V^A - V^*||_\infty$ aber von der Höhe des maximalen Sprungs in der Wertfunktion ab, die von dem eingesetzten Approximationsschema A nicht korrekt erfasst werden kann. Verwendet ein Approximator aufgrund einer ungünstigen Merkmalsraumabbildung in der Mittelung weit voneinander entfernte Observationen mit völlig unterschiedlichen Zustandswerten, so wie im Beispiel in Abbildung 4.4, ist dieser Fehler auch für Observationen-MDPs mit glatter optimaler Wertfunktion hoch und die Aussagekraft dieser Schranke bezüglich des Abstandes von $\bar{V}' : O \mapsto \mathbb{R}$ zur optimalen Zustandswertfunktion $V^* : O \mapsto \mathbb{R}$ noch geringer als sonst (vgl. erster Punkt in Aufzählung auf Seite 59). Auch die von Gordon diskutierte Idee [53, Abschnitt 6, Seiten 13–14], für glatte (Lipschitz-stetige) Wertfunktionen die Zellen einer Partitionierung zu verringern und so den maximalen Fehler in einer Zelle zu verkleinern [25, 26], funktioniert nicht, da eine Verringerung der Zellgröße auf Z keine solche Bedeutung auf den Observationen hat – ganz davon abgesehen, dass die Wertfunktionen auf den Observationen-MDPs in den seltensten Fällen stetig sind. Hieraus ergeben sich auch für die Konsistenz des Algorithmus erhebliche Probleme.

4.5 Zur Konsistenz der äußeren Schleife

Neben der Stabilität des in der inneren Schleife vollzogenen batch RL-Vorgangs stellt sich außerdem die Frage nach der Entwicklung der mittels FQI erlernten Wertfunktionen über mehrere Iterationen der äußeren Schleife hinweg. Hier sind im Falle von DFQ zwei Aspekte zu betrachten:

1. Verhält sich die innere Schleife von DFQ konsistent im Sinne von Lemma 5, wenn in den Explorationsphasen der äußeren Schleife zusätzliche Daten gesammelt werden?

2. Welchen Einflus hat der Generationswechsel auf die Qualität der Wertfunktionen, die vor und nach der einhergehenden Änderung des Merkmalsraums erlernt wurde?

4.5.1 Stochastische Konsistenz über Explorationsphasen hinweg

Motivation für den Wechsel zwischen Strategieverbesserung und Interaktion mit dem System innerhalb von DFQ und anderen semi-online Verfahren ist die bessere Exploration des Systems. In Hinblick auf den von Ormoneit und Sen aufgeworfenen Aspekt der Konsistenz stellt sich für DFQ nun die Frage, ob sich über die Intuition hinaus

formal beweisen lässt, dass die zusätzliche Exploration zu einer Verbesserung der in der inneren Schleife berechneten Wertfunktionen führt; sich diese vielleicht sogar im Limes der optimalen Wertfunktion nähert.

Leider kann die Anwendung der von Ormoneit und Sen erarbeiteten Aussagen und Werkzeuge auf DFQ in dieser Hinsicht nicht zum Erfolg führen. Die stochastische Konsistenz nach Ormoneit und Sen beruht nach Lemma 4 auf der stetigen Absenkung der Bandbreite des Kernels nach einer zulässigen Rate. Die Verkleinerung des Einzugsbereichs auf den Lipschitz-stetigen Funktionen spielt eine große Rolle im Beweis. Wie schon festgestellt, kann der für die Observationen abgeleitete Kernel k' aber unter der nichtlinearen Abbildung zerrissen werden, so dass k' weder (4.36) noch der Forderung nach Lipschitz-Stetigkeit [113, Anhang A.1] genügt, selbst wenn k ein passender Kernel ist. Zudem bedeutet eine Absenkung der Bandbreite des vorgegebenen Kernels k nicht wie gefordert, dass sich auch der Einzugsbereich des Kernels k' im Observationen-MDP gleichermaßen verkleinert. Eine formale Untersuchung auf Z scheidet wie gehabt aus, da die Merkmalsvektoren nicht markov sind und Wertfunktionen eine hypothetische Stetigkeit durch die nichtlineare Abbildung auf den Merkmalsraum mit sehr hoher Wahrscheinlichkeit verlieren.

In der Praxis ergibt sich zudem das nicht nur auf den Einsatz von DFQ beschränkte Problem, dass interessante Wertfunktionen – insbesondere von zeitdiskreten Systemen – praktisch nie Lipschitz-stetig sind – weder auf den Systemzuständen noch den Observationen noch den Merkmalen. Ormoneit und Sens Konsistenzergebnis kommt daher eher theoretische Bedeutung zu, auch wenn es mit der Vergrößerung der Stichprobe und einer geeigneten Erhöhung der Auflösung des Approximators sicher sinnvolle Anhaltspunkte für eine Verbesserung einer erlernten Wertfunktion gibt.

4.5.2 Einfluss der Generationswechsel

Wird in der äußeren Schleife ein neuer Encoder trainiert und ändern sich deshalb der Merkmalsraum und die vom Encoder realisierte Abbildung $\mathrm{ENC} : O \mapsto Z$ der Observationen auf die Merkmalsvektoren, ist dies gleichbedeutend mit einer Änderung des zu k korrespondierenden Kernels k'. Auch wenn sich der durch $k : Z \times Z \mapsto R$ definierte Anteil des Kernels $k'(o, o') = k(\mathrm{ENC}(o; W), \mathrm{ENC}(o'; W))$ nicht ändert, weil k auch nach der Änderung der Einbettung weiterverwendet wird, ändert sich doch der nichtlineare, vom Encoder definierte Anteil mit jedem Generationswechsel. D.h., selbst wenn nach einem Generationswechsel mit dem gleichen Funktionsapproximationsschema in FQI weitergearbeitet wird, ist der Zufallsoperator $\widetilde{H}^a_{dp}$ nicht mehr derselbe wie vor dem Generationswechsel. Es ist nicht ausgeschlossen, dass sich durch diese Veränderung die in der inneren Schleife berechnete Wertfunktion anschließend verschlechtert. Die Anwendung von Ormoneit und Sens Konsistenzergebnissen über einen Generationswechsel hinweg scheidet aus, denn in Lemma 8 und Satz 4 wird von der Anwendung eines konstanten Zufallsoperators ausgegangen.

Tatsächlich ist auch nicht sichergestellt, dass sich die Einbettung bei einem Generationswechsel wirklich verbessert. Je nach Initialisierung der Gewichte des Autoencoders landet man bei unterschiedlichen Ergebnissen von schwankender Qualität. Üblicherwei-

se wird deshalb von mehreren Versuchen der beste Encoder ausgewählt [61]. Es fehlen zudem Werkzeuge zur formalen Untersuchung der Encoder, die helfen würden, die zu erwartende Eignung erzeugter Merkmalsräume für das Erlernen einer Strategie – z.B. durch eine Schranke – einzugrenzen.

4.6 Diskussion der Ergebnisse

Für die Konvergenz in diskontierten Problemstellungen konnte ein starkes Ergebnis erzielt werden. Der FQI-Algorithmus in der inneren Schleife von DFQ konvergiert zu einem eindeutigen Fixpunkt, obwohl ihm eine nichtlineare Abbildung der Observationen auf die als Basis verwendeten Merkmalsvektoren vorgeschaltet wurde. Das ist eine wichtige Aussage für die praktische Anwendung von DFQ. Aufgrund der Eindeutigkeit der Lösung, unabhängig vom Startpunkt, belegt es auch die Gefahrlosigkeit des Einsatzes der in Kapitel 3 eingeführten Technik zum Übersetzen der Wertfunktion.

Die darüber hinaus für DFQ erzielten Ergebnisse lassen sich in einem pointierten Satz zusammenfassen: Alles steht und fällt mit der Qualität des erzeugten Merkmalsraums. Unabhängig von seiner Qualität ist zwar die Konvergenz für diskontierte Observationen-MDPs sichergestellt, aber die Konvergenz in kürzester Pfad Problemen, die Qualität der Lösungen im Vergleich zur optimalen Wertfunktion, die Konsistenz des Ergebnisses über mehrere Iterationen der äußeren Schleife hinweg und die Auswirkungen eines Generationswechsels hängen alle von den Eigenschaften der Einbettung ab. Im Detail:

Fehlerschranke Die von Gordon hergeleitete obere Schranke gilt prinzipiell auch, wenn man FVI auf den Merkmalsraum anwendet. Sie hat in der Praxis aber nur eine geringe Aussagekraft, insbesondere deshalb, weil von der nichtlinearen Transformation bestehende Nachbarschaftsbeziehungen auseinander gerissen werden können. Nur wenn Merkmalsvektoren ähnlicher Zustände in unmittelbarer Nachbarschaft landen und die Stetigkeit einer Wertfunktion unter der Abbildung erhalten bleibt, kann mit den von einem auf Z arbeitenden Approximator erfassten Nachbarschaften und den dadurch maximal eingeführten Fehlern – z.B. im Sinne des Lipschitz-Faktors L – argumentiert werden.

Nicht diskontierte MDPs Ob die Folge der erlernten Wertfunktionen in nicht diskontierten MDPs konvergieren kann, hängt davon ab, ob der Encoder das MDP zerreisst oder Verdeckungen erzeugt. Durch die nichtlineare Einbettung entstehen zusätzliche Probleme bei der Überprüfung der Kompatibilität eines gegebenen Funktionsapproximators. Es gibt zwei Lösungsansätze: A) Vorsichtig zu sein und grundsätzlich $\gamma < 1$ zu wählen. Der Lösungsvorschlag, alle Gewichte $\beta_{ij} > 0$ zu wählen, reicht jetzt nicht mehr, da auch der nichtlineare Encoderanteil im Kernel k' bereits für irreparable Inkompatibilität sorgen kann. B) Zu versuchen, die notwendigen Kriterien auf Z selbst herzustellen, so dass die Untersuchung des Kernels k allein im Merkmalsraum vorgenommen werden kann. Insbesondere muss die Abbildung ENC : $O \mapsto Z$ hierfür markovsche Merkmalsvektoren erzeugen und darf

somit keine Zustände verdecken.

Konsistenz Die Ergebnisse von Ormoneit und Sen legen nahe, dass die Einbeziehung von Exploration in den batch RL-Ablauf prinzipiell Sinn ergibt. Man kann sich berechtigte Hoffnungen machen, dass sich allein schon durch die Vergrößerung der Stichprobe in der Erwartung auch die Güte der errechneten Wertfunktionen verbessert. Unter strengen Annahmen lässt sich dies sogar für eine Klasse von glatten Wertfunktionen formal beweisen. In der Praxis sollte genauso die bereits von Gordon aufgegriffene und von Ormoneit und Sen perfektionierte Idee von Chow und Tsitsiklis [26] zur Anwendung kommen, die Auflösung des Approximators mit zunehmender Stichprobengröße zu verfeinern.

Ein formaler Nachweis der Konsistenz erfordert allerdings, dass sich die Auflösung des eingesetzten Kernels bei einer gleichzeitigen Glattheit der optimalen Wertfunktion verkleinern lässt. Beides ist in der betrachteten Problemstellung beim Einsatz eines tiefen Encoders in der Regel nicht gegeben. Sind die Merkmalsvektoren so angeordnet, dass sie sehr gut Nachbarschaften des Observationen-MDPs, oder besser noch des Zustand-MDPs widerspiegeln, ist diese Glattheit zumindest eher gegeben, als wenn das Ursprungs-MDP durch den Encoder zerrissen und die Zustände chaotisch angeordnet werden.

Generationswechsel Die Generationswechsel führen zu einer neuen Einbettung der Observationen in den Merkmalsraum und damit zu einem veränderten Zufallsoperator $\hat{H}^a_{dp}$. Ob dieser besser oder schlechter als der Zufallsoperator vor dem Generationswechsel ist, hängt vornehmlich von der Eignung der vom neuen Encoder erzeugten Merkmalsvektoren ab. Bei der Verwendung tiefer Autoencodernetze kommt es hier in jedem Fall zu Schwankungen, was ein weiteres Argument für die in Kapitel 3 diskutierten Methoden zur Reduzierung der Anzahl von Generationswechseln ist.

Ganz offensichtlich müssen für die Beurteilung der "Qualität" eines erlernten Merkmalsraums drei Kriterien im Sinne der vorhergehenden Diskussion herausgestellt werden.

1. Die Merkmalsvektoren müssen Systemzustände eindeutig idenitifizieren und somit markov sein. In diesem Fall ließe sich gewährleisten, dass ein Merkmals-MDP aufgestellt und direkt untersucht werden kann. Gleichzeitig wären Verdeckungen von Zuständen ausgeschlossen.

2. Merkmalsvektoren ähnlicher Zustände müssen in unmittelbarer Nachbarschaft liegen. Nur wenn umliegende Merkmalsvektoren einen ähnlichen Zustandswert besitzen, macht die Diskussion über die Zellgröße bzw. die Bandbreite eines Kernels Sinn und nur dann ist es möglich, den Fehler, der durch das Heranziehen der Werte ähnlicher Zustände entsteht, zum Beispiel in Form einer Lipschitz-Konstante abzuschätzen. Auch intuitiv macht der Einsatz der Funktionsapproximation auf dem Merkmalsraum im Sinne der Mittelung über benachbarte Samples nur dann Sinn,

wenn unter der eingesetzten Metrik benachbarte Zustände sich mit hoher Wahrscheinlichkeit auch hinsichtlich optimalen Zustandswertes und optimaler Aktion ähneln.

3. Etwaige Diskontinuitäten und Entscheidungsgrenzen müssen im Merkmalsraum erkennbar und vom Funktionsapproximator erfassbar sein. Schon Gordon hat die Wichtigkeit des richtigen Erfassens von vorhandenen Diskontinuitäten und Entscheidungsgrenzen für die zu erreichende Qualität der approximierten Wertfunktion herausgestellt (siehe Abschnitt 4.1.2). In den Konsistenzaussagen für KADP spiegelt sich dies in gewisser Weise im Lipschitz-Faktor wieder.

Bei einer solchermaßen zentralen Bedeutung wäre die Möglichkeit einer formalen Analyse der innerhalb von DFQ erzeugten Merkmalsräume hinsichtlich dieser Kriterien wünschenswert. Leider stellt dies eine nicht zu überwindende Hürde dar. Prinzipiell konvergieren die Autoencoder als MLPs je nach Initialisierung der Gewichte zu unterschiedlichen lokalen Minima und erzielen somit bei jeder Neuberechnung unterschiedliche Ergebnisse. Da die Kodierungsschicht der tiefen Autoencoder, die den Merkmalsraum erzeugt, nicht Gegenstand eines expliziten Optimierungskriteriums ist – trainiert wird auf die Minimierung des Rekonstruktionsfehlers in der äußeren Schicht – fehlt zudem jede Möglichkeit, formale Aussagen zur Güte oder zu anderen Eigenschaften der erzielten Kodierungen zu machen. Und schon gar nicht ist unter formalen Gesichtspunkten klar, ob eine kontinuierliche Verringerung des Rekonstruktionsfehlers notwendigerweise zu einem Merkmalsraum führt, der besser für die Lösung eines individuellen Reinforcement Lernproblems geeignet ist. An dieser Stelle besteht also im Moment eine grundsätzliche formale Lücke, die den Rückgriff auf empirische Untersuchungen erfordert.

4.7 Zusammenfassung

In diesem Kapitel wurden die grundlegenden theoretischen Ergebnisse zum batch RL ausführlich wiedergegeben, dabei kleinere Lücken geschlossen und die Resultate von Gordon, Ernst und Ormoneit et al. in einen engeren Zusammenhang gestellt. FQI wurde bei allen von Ernst hervorgehobenen Unterschieden als Variante des KADP-Verfahrens identifiziert. Die von Gordon und Ormoneit et al. getroffenen Annahmen und erzielten Aussagen wurden ausführlich hinsichtlich ihrer Implikationen für praktische Anwendungen von batch RL diskutiert. Auf Basis dieses theoretischen Fundaments wurde dann die Konvergenz und Konsistenz von DFQ analysiert.

Zunächst erscheint die nichtlineare Abbildung der Observationen in den Merkmalsraum einer stabilen Konvergenz der inneren Schleife von DFQ im Wege zu stehen. Tatsächlich lässt sich die Einbettung aber als Teil des eingesetzten Funktionsapproximators auffassen, so dass man sie in den Gewichten des Approximators, bzw. in der Kernelfunktion “verschwinden” lassen und die nachweisliche Konvergenz für diskontierte Problemstellungen sicherstellen kann. Andererseits kann es passieren, dass ähnliche Zustände des Systems unter der nichtlinearen Einbettung auf völlig unterschiedliche Merkmalsvektoren abgebildet werden. In einem solchen Fall ist es nicht möglich, wie in

den Konsistenzbeweisen gefordert, den Einflussbereich der Kernel in geeigneter Weise über den Bandbreitenparameter abzusenken und die Stetigkeit der Kernel bzw. Wertfunktionen sicherzustellen. Auch kann es zu Verdeckungen von Zuständen kommen, was zur Divergenz auf nicht diskontierten Problemstellungen führen kann.

Eine formale Analyse der Eigenschaften der erzeugten Merkmalsräume, die für weitere positive Aussagen nötig wäre, scheitert daran, dass ein Optimierungskriterium und hilfreiche formale Aussagen für die Kodierungsschicht der Autoencoder gänzlich fehlen. Diese formale Lücke in DFQ gibt Anlass zur Duchführung von ausführlichen empirischen Untersuchungen in Kapitel 5. Aus theoretischer Sicht sind hier drei Forderungen zu stellen: Unterschiedliche Zustände müssen an Hand der Merkmalsvektoren identifizierbar sein, ähnliche Zustände müssen benachbart sein, Entscheidungsgrenzen und Diskontinuitäten müssen erfassbar sein.

Neben den Merkmalsräumen kommt auch dem in FQI eingesetzten Funktionsapproximator eine wichtige Bedeutung bezüglich der Stabilität von DFQ zu. Wegen der tiefen Netze zur Merkmalsraumerzeugung liegt eigentlich der Einsatz von MLPs wie in NFQ nahe. Allerdings würde die Verwendung von Nicht-Averagern zu einer nachweislichen Instabilität der inneren Schleife von DFQ und damit zu einer zusätzlichen Ursache für Schwankungen im Lernverlauf führen – neben der Verwendung der Merkmalsräume und der Generationswechsel. Ein weiterer kritischer Aspekt ist die Auswahl möglichst kompatibler Approximatoren für die erlernten Abbildungen, ohne dass die genauen Eigenschaften der erzeugten Merkmalsräume vorab bekannt wären. Diese Aspekte werden in Kapitel 6 mit dem eigens für DFQ entwickelten ClusterRL Verfahren aufgegriffen.

5

Automatische Konstruktion von Merkmalsräumen in DFQ

In diesem Kapitel wird die Verwendung tiefen Lernens zur Abbildung von Bilddaten in einen niedrigdimensionalen, sehr kompakten Merkmalsraum beschrieben und die Tauglichkeit dieser Methode für eine Anwendung in der äußeren Schleife von DFQ untersucht. Im Mittelpunkt des ersten Teils dieses Kapitels stehen dabei die Beschreibung des in DFQ konkret eingesetzten Lernverfahrens mit tiefen Autoencodern, das sich in einigen Details von bestehenden Verfahren abhebt, und die praktische Umsetzung in einer Rechenzeit einsparenden, parallelen Implementierung. Der zweite Teil ist dann ganz der empirischen Untersuchung der von den tiefen Autoencodern erzeugten Merkmalsräume gewidmet. Dabei werden auch die im vorhergehenden Kapitel 4 aufgestellten Anforderungen berücksichtigt und in den verwendeten Bewertungskriterien aufgegriffen. Es werden einige praktische Probleme beim Einsatz der tiefen Autoencoder identifiziert. Mit den rezeptiven Feldern und Faltungskernen in den äußeren Schichten der Autoencoder werden Methoden entwickelt und erprobt, die gerade im Einsatz innerhalb von DFQ helfen, die erzeugten Merkmalsräume zu verbessern.

5.1 Motivation

In vielen der bisher zum tiefen Lernen publizierten Artikel wurden im Bereich der Bildverarbeitung vornehmlich klassische Objektklassifikations- und Erkennungsaufgaben behandelt [12, 61]. Tiefe Netze dienen hier der Extraktion von Merkmalen, die eine Unterscheidung und Klassifikation der meist in den Bildausschnitten zentrierten Objekte (Gesichter verschiedener Datensätze [24, 61], handgeschriebene Zahlen des MNIST-Datensatzes [88]) ermöglichen [24, 61, 83, 84, 106, 144]. Die erzeugten Merkmalsvektoren besitzen in den Klassifikationsaufgaben wenige Dutzend [144] bis hin zu einigen Hunderten oder Tausenden Dimensionen [61], auf deren Basis dann eine Ausgabeschicht überwacht auf die unär kodierten Klassenlabel trainiert [61] oder direkt eine Nächster-Nachbar-Klassifikation [144] angewendet wird.

Nicht untersucht wurden in diesen Arbeiten dagegen Problemstellungen, bei denen

ein oder mehrere Objekte an völlig verschiedenen Stellen im Bild auftauchen können und geortet werden müssen, so wie es in den im Rahmen dieser Arbeit untersuchten Navigationsaufgaben vorkommt[1]. Auch wurde in Hintons Arbeiten weitesgehend der Effekt des Bildrauschens ausgeblendet, obwohl sich Pixelrauschen beim Einsatz einer realen Foto- oder Videokamera nicht vermeiden lässt und eine zusätzliche Schwierigkeit bei der Klassifikation darstellt. Eine offene Frage ist daher, ob es mit Hilfe der tiefen Netze auch gelingen kann, unüberwacht zu erlernen, zunächst die Position eines relevanten Objektes aus Bildern robust zu detektieren und diese dann in einer sehr kompakten Merkmalsschicht mit nur ein oder zwei Dimensionen (in Anlehnung an die x/y-Bildkoordinaten) zu kodieren. Eine solch kompakte Einbettung ist notwendig, um auf niedrigdimensionale Räume beschränkte Reinforcement Lernverfahren zum Einsatz bringen zu können.

Neben der extrem niedrigen Dimensionalität des Merkmalsraums ergeben sich aus der Kombination des tiefen Lernens mit einem RL-Ansatz einige weitere besondere Anforderungen an die erzeugten Merkmalsräume, die in der Folge näher erläutert werden sollen:

1. Identifizierung, Differenzierung und Generalisierung Der Erfolg des Strategielernens auf einem Merkmalsraum hängt insbesondere davon ab, wie gut die erzeugten Merkmalsvektoren Rückschlüsse auf den nicht beobachtbaren Systemzustand erlauben. Aus theoretischer Sicht wurde die grundlegende Bedeutung dieses Aspekts für die Konvergenz und Konsistenz des DFQ-Algorithmus bereits in Kapitel 4 herausgestellt. So sollten zwei Bilder, die ein Objekt an der gleichen Position zeigen, zu einem ähnlichen Merkmalsvektor führen, der sich von den Vektoren anderer Objektpositionen deutlich unterscheidet. Insbesondere auch das Rauschen, das im Bildformationsprozess typischerweise auftritt, sollte nicht so leicht zu Fehlern führen. Ist eine Unterscheidung nicht eindeutig möglich, dann kommt es zu "Zustandsverdeckungen" oder "Zustandsaliasing", also zu Verwechselungen des internen Systemzustandes, die sich nicht mehr durch das nachgeschaltete Strategielernen ausgleichen lassen und somit zwangsläufig zu Verschlechterungen gegenüber der optimalen Strategie führen. Da die den Algorithmen zugrunde liegende Markov-Eigenschaft in diesem Fall verletzt wird, kann es zu Instabilitäten bis hin zur Divergenz des Lernvorgangs kommen.

Neben dieser zustandsidentifizierenden Eigenschaft der Merkmalsräume spielt aber auch die Anordnung von Bildern unterschiedlicher Systemzustände im Merkmalsraum eine Rolle. Bilder ähnlicher Systemzustände müssten sich auch im Merkmalsraum in unmittelbarer Nachbarschaft wiederfinden, damit lokal generalisierende RL-Algorithmen aus dieser Anordnung einen Nutzen ziehen können und nicht jede mögliche Observation beobachten und auswendig lernen müssen. Diesem Aspekt kommt laut der Diskussion in Kapitel 4 auch eine wichtige theoretische Bedeutung zu, wenn es um den Einsatz eines

[1] Eine Ausnahme bildet [124], wo eine mehrschichtige Architektur für translationsinvariante Klassifikation vorgestellt wird. Diese Publikation beschränkt sich allerdings weiterhin auf die Klassifikation, nicht Lokalisation, der Objekte und setzt eine sehr spezielle, nicht durchweg auf neuronalen Netzen basierende Architektur mit einem EM-Trainingsverfahren ein.

KADP-Verfahrens und die Konsistenz des DFQ-Algorithmus geht. Gleichzeitig muss die erlernte Anordnung es einem Funktionsapproximator erlauben, lokale Unterschiede in der optimalen Wertfunktion, Diskontinuitäten und Entscheidungsgrenzen adäquat im Merkmalsraum zu erfassen, um grundsätzliche Kompatibilität zum MDP zu ermöglichen.

Aufgrund dieser Überlegungen lassen sich unter Einbeziehung der in Abschnitt 4.7 genannten, aus theoretischer Sicht relevanten Anforderungen folgende Kriterien für die Bewertung der durch das tiefe Lernen erzeugten Merkmalsräume formulieren:

Kriterium 1: Zustandsidentifizierende Eigenschaft Observationen, denen unterschiedliche Systemzustände zugrunde liegen, müssen auch im Merkmalsraum an verschiedenen Stellen liegen und möglichst klar voneinander trennbar sein. Zustandsverdeckungen sind zu vermeiden.

Kriterium 2: Robustheit Merkmalsvektoren mehrerer verrauschter Bilder, die eine exakt identische Situation zeigen, sollten im Merkmalsraum sehr dicht beieinander liegen und gut von Merkmalsvektoren anderer Situationen separierbar bleiben. In Bezug auf die in Abschnitt 3.2 erarbeitete Modellierung bedeutet dies, dass die unter der linkseindeutigen Bildformation entstehenden, verschiedenen Observationen ein und desselben Systemzustands im Idealfall auf den gleichen bzw. sehr eng beieinander liegende Merkmalsvektoren abgebildet werden.

Kriterium 3: Nachbarschaft ähnlicher Zustände Zwei Bilder, die eine ähnliche Situation zeigen – z.B. ein Objekt an nur leicht voneinander abweichenden Positionen – sollten nicht zu stark voneinander abweichenden Merkmalsvektoren führen, sondern die Tendenz haben, in benachbarten Clustern – wenn nicht sogar im selben Cluster – zu liegen. Gleichzeitig muss es möglich sein, Merkmalsvektoren mit sehr unterschiedlichen optimalen Zustandswerten voneinander zu separieren.

Kriterium 4: Übertragung der Topologie Eine gute Abbildung der Topologie – im Sinne der globalen Anordnung der zugrunde liegenden Systemzustände – auch im Merkmalsraum kann beim Lernen der Strategie helfen, wenn dort Approximatoren mit der Fähigkeit zum globalen Generalisieren eingesetzt werden.

Der Grad der Einhaltung dieser Kriterien beeinflusst ganz maßgeblich die Qualität resultierender Strategien. Kriterium 1 wirkt sich vornehmlich auf die Stabilität, Kriterien 2 und 3 wirken sich auch auf die Generalisierung und Dateneffizienz approximierender batch RL-Ansätze aus. Kriterium 3 ist zudem eine wichtige Voraussetzung, um über die Konsistenz argumentieren zu können (siehe Kapitel 4). Die Erhaltung der globalen Topologie ist dagegen kein notwendiges, sondern ein optionales Kriterium, das ausschließlich von globalen Approximatoren positiv ausgenutzt werden kann.

2. Umgang mit kleinen, ungleichmäßig verteilten Trainingsdatensätzen Während der Explorationsphasen in DFQ kann, anders als in den in der Literatur untersuchten Objektklassifikationsaufgaben, an vielen realen Systemen nicht sichergestellt werden, dass sich die beobachteten Observationen gleichmäßig über den Observationsraum

verteilen. Typischerweise kommt es zu Beginn des Lernens um die Startzustände und später um die Zielzustände zu Häufungen. Oftmals führen die optimalen Trajektorien, entlang derer sich der Agent mit zunehmendem Lernfortschritt immer häufiger bewegt, auch nur durch einen Teil des Zustands- bzw. Observationsraums und decken ihn nicht vollständig ab.

Erschwerend kommt hinzu, dass beim Reinforcement Lernen an realen Systemen meist wesentlich weniger Observationen als zum Beispiel im MNIST-Datensatz (6 000 Bilder je Klasse, insgesamt 60 000 Bilder, davon 50 000 zum Training eingesetzt [61]) zur Verfügung stehen. Die Frage ist, ob das tiefe Lernen trotz dieser ungleichmäßigen Verteilung und der deutlich geringeren Zahl an Trainingsbeispielen ähnliche gute Einbettungen wie in den bisherigen Untersuchungen erzeugt und zu überzeugender Generalisierung führt. Gerade zu Beginn des Lernens ist es wichtig, bereits aus wenigen verfügbaren Observationen eine frühe Einbettung ableiten zu können, um eine erste Strategie zu erlernen und die Exploration in interessante, noch nicht untersuchte Gebiete voranzutreiben.

Generell wurde dieser Aspekt in der Literatur bisher nur wenig untersucht. Der Schwerpunkt wurde auf den Vergleich von Lernmethoden und die Auswahl und Vorteile einzelner Topologien gelegt. Ausreichende, alle Klassen gleichmäßig repräsentierende Datensätze wurden dabei meist vorausgesetzt. Aus diesen Gründen ist es an dieser Stelle umso wichtiger, Untersuchungen nicht nur auf großen Datensätzen mit perfekter Abdeckung durchzuführen. Gute Ergebnisse hinsichtlich der oben genannten vier Kriterien werden vielmehr auch auf kleinen Datensätzen mit schlechter und / oder ungleichmäßiger Abdeckung benötigt. Als Kriterium für die Schwierigkeit eines Trainingsdatensatzes kann die – in der Folge als “Abdeckung” bezeichnete – Anzahl der Trainingsbeispiele pro möglicher Bildposition (“Klasse” in den Navigationsexperimenten) herangezogen werden. Bei Ernst betrug das Verhältnis 10/1 [40], bei Gordon war es größer als 25/1 [51] und im nicht ganz vergleichbaren MNIST-Problem werden üblicherweise pro Klasse sogar 5 000 Trainingsbilder verwendet.

Bevor entsprechende Untersuchungen zu diesen zwei Aspekten durchgeführt werden, wird in den folgenden Abschnitten zunächst die verwendete Methode zum Training von tiefen Autoencodernetzen und deren effiziente Implementierung beschrieben. Hier ist ausnahmsweise nicht die Dateneffizienz, sondern die Rechenzeit gemeint, da die Reduzierung der Laufzeit von einigen Tagen zu wenigen Stunden eine ganz entscheidende Rolle bei der späteren Anwendung auf reale Systeme darstellt (siehe Kapitel 7).

5.2 Training mit flachen Autoencodern

Beim Studium der Literatur kann leicht der Eindruck entstehen, als seien tiefe Autoencoder und die insbesondere von Hinton beworbenen RBMs untrennbar miteinander verknüpft [60, 61, 144, 145]. Bengio weist aber in [12] zurecht darauf hin, dass das tiefe Lernen nicht nur auf das Training solcher RBMs beschränkt ist, sondern wesentlich weitergefasst werden kann und auch andere Methoden des Trainings vielschichtiger

Architekturen beinhaltet – nicht einmal nur neuronaler Netze.

In [12, Kapitel 9] beschreibt Bengio, wie tiefe Encodernetzwerke Schicht für Schicht durch das aufeinander folgende Training einzelner einschichtiger Autoencodernetze auch ohne RBMs aufgebaut werden können. In [11, 84, 168] wird das Vorgehen deutlicher: Es kommt eine spezielle Form der flachen Autoencodernetze zum Einsatz, bei denen die Gewichte $w_{ji} \in W_1$ von der Eingabe zur versteckten Schicht identisch mit den entsprechenden Gewichten $w_{ij} \in W_2$ von der versteckten zur Ausgabeschicht sind bzw gilt: $W_2 = W_1^T$. Optimiert werden die Gewichte über einen Gradientenabstieg auf dem Kreuzentropiefehler der Rekonstruktionen der Eingabe. Die Aktivierungen der Neuronen werden dabei in der Anwendung auf binärwertigen Eingabedaten weiterhin als Wahrscheinlichkeit für das Feuern des entsprechenden Neurons interpretiert.

In von Larochelle, Bengio und anderen durchgeführten Vergleichen [83, 84, 168] wurden bei Verwendung flacher Autoencoder und Backpropagation im Vortraining zu Hintons RBMs vergleichbare Ergebnisse erzielt, wobei je nach konkreter Problemstellung die mittels RBMs trainierten Netze [83, 84], in anderen Problemstellungen die reinen MLPs [84, 168] leichte Vorteile hatten. In [168] wurden von Vincent et al. weitere Methoden aufgezeigt, um die Nachteile der MLPs auszugleichen, die in manchen Problemstellungen beobachtet wurden. Es spricht also grundsätzlich nichts gegen die Verwendung von MLPs und Backpropagation im Vortraining an Stelle der RBMs und Contrastive Divergence.

Der Ablauf des in DFQ eingesetzten Trainingsalgorithmus ist prinzipiell der gleiche wie in den von Hinton beschriebenen Experimenten [61]. Es gibt ein schichtenweises Vortraining und eine anschließende Finetuningphase des gesamten Netzes (siehe Abschnitt 2.1.4). In einer Variation des von Bengio et al. beschriebenen Trainings [11, 12, 84] kommen in DFQ statt der recht speziellen RBMs aber auch im Vortraining ausschließlich ganz gewöhnliche MLPs zum Einsatz, die hier mittels RProp trainiert werden.

5.2.1 In DFQ verwendeter Algorithmus

Es wird angenommen, dass ein tiefer Autoencoder mit je $h > 0$ versteckten Schichten vor und nach der Kodierungsschicht, also mit insgesamt $2h + 3$ Schichten, auf einer Menge von normalisierten, k_1-dimensionalen Eingabedaten $\mathcal{X}^0 \subset [0,1]^{k_1}$ trainiert werden soll. Die Anzahl der Neuronen in Schicht $i = 1, \ldots, h+2$ und Schicht $2h+4-i$ sei gegeben durch k_i. Das Vortraining der Verbindungsgewichte wird von außen nach innen, von Eingabe- und Ausgabeschicht hin zur Kodierungsschicht, vorgenommen (siehe auch Abb. 5.1). Es wird mit dem Training der Verbindungsgewichte von der ersten zur zweiten Schicht und von der $2h-4-1$-ten (Ausgabeschicht) zur $2h-4-2$-ten Schicht (erste versteckte Schicht von hinten) begonnen und läuft im Detail wie folgt ab:

---

1. Zunächst wird ein dreischichtiges Netz mit $k_i = k_{2h+4-i}$, $k_{i+1} = k_{2h+3-i}$ und $k_{2h+4-i} = k_i$ Neuronen in Eingabeschicht, versteckter Schicht und Ausgabeschicht konstruiert. Es werden in diesem flachen Autoencoder exakt die gleichen Verbindungen wie zwischen den entsprechenden Schichten des tiefen Autoencoders (sie-

he Abbildung 5.1) erzeugt und mit $\mathcal{N}(0, \sigma_0)$ zufällig initialisiert. Abweichend von [61, 84] sind die Gewichte W^1 der ersten und W^2 der zweiten Schicht des erzeugten flachen Autoencoders nicht voneinander abhängig, insbesondere gilt auch nicht $w^1_{ji} = w^2_{ij}$.

2. Der flache Autoencoder wird nun auf den vorliegenden Eingaben mittels Gradientenabstiegs auf die Identitätsfunktion trainiert. Zum Training wird RProp verwendet. In der Praxis hat sich erwiesen, dass es nicht notwendig ist, das Netz ganz auszutrainieren. Übertraining sollte aber in jedem Fall durch geeignete Regularisierungsmaßnahmen (z.B. Early Stopping, Weight Decay) verhindert werden.

3. Die vortrainierten Gewichte werden vom flachen Autoencoder an die richtige Stelle im tiefen Autoencoder zurückgeschrieben.

4. Die Eingabedaten $\mathcal{X}^i$ werden noch einmal durch den flachen Encoder propagiert. Dabei wird die Aktivierung der versteckten Neuronen ausgelesen und als neuer Datensatz $\mathcal{X}^{i+1}$ gespeichert.

5. Nun wird mit dem Training der nächstinneren Schicht in Schritt 1 fortgefahren. Zum Training wird der Datensatz $\mathcal{X}^{i+1}$ verwendet, der Zähler wird um Eins erhöht $i \leftarrow i + 1$.

Im tiefen Autoencoder summieren sich die Rekonstruktionsfehler, die nach jedem Vortraining eines flachen Autoencoders verbleiben, von Schicht zu Schicht auf. Allein deshalb wird nach Abschluss des schichtenweisen Vortrainings der tiefe Autoencoder in der Feinabstimmungsphase für einige Epochen als gesamtes Netz mit RProp weiter trainiert.

Der aufgrund des Einsatzes gewöhnlicher MLPs mögliche Verzicht auf die Forderung nach symmetrischen Verbindungsgewichten ermöglicht es, gänzlich unterschiedliche Verbindungsstrukturen zwischen den korrespondierenden Schichten des Encoders und Decoders zu verwenden. Die Beschränkung auf ein gemeinsames Trainingsverfahren beim Vortraining und Finetuning kann als weitere Erleichterung angesehen werden, zumal MLPs mit ihren Trainings- und Regularisierungsmethoden im Allgemeinen besser verstanden und breiter untersucht sind als die recht speziellen RBMs. Das sowohl zum Vortraining als auch beim Finetuning verwendete Resilient Propagation hat sich vielfach als sehr robust hinsichtlich der Parameter erwiesen und konvergiert vergleichsweise schnell (siehe Absatz "Resilient Propagation" im Abschnitt 2.1.1). Dabei weist es im Vergleich zu den von Hinton eingesetzten [61], auch sehr leistungsfähigen konjugierten Gradientenverfahren einen deutlich geringeren Berechnungsaufwand auf.

Das beschriebene Verfahren stellt somit die unmittelbare Übertragung der beiden wirklich zentralen Ideen der tiefen Autoencoder – das schichtenweise Vortraining und die Durchführung der hochdimensionalen Daten durch einen engen Flaschenhals – auf "gewöhnliche" MLPs dar, ohne die eher ungewöhnlichen Besonderheiten und speziellen Netzarchitekturen wie die Verwendung von stochastischen Aktivierungen [61] und symmetrischen Gewichten zwischen den Schichten [11, 61, 84] beizubehalten.

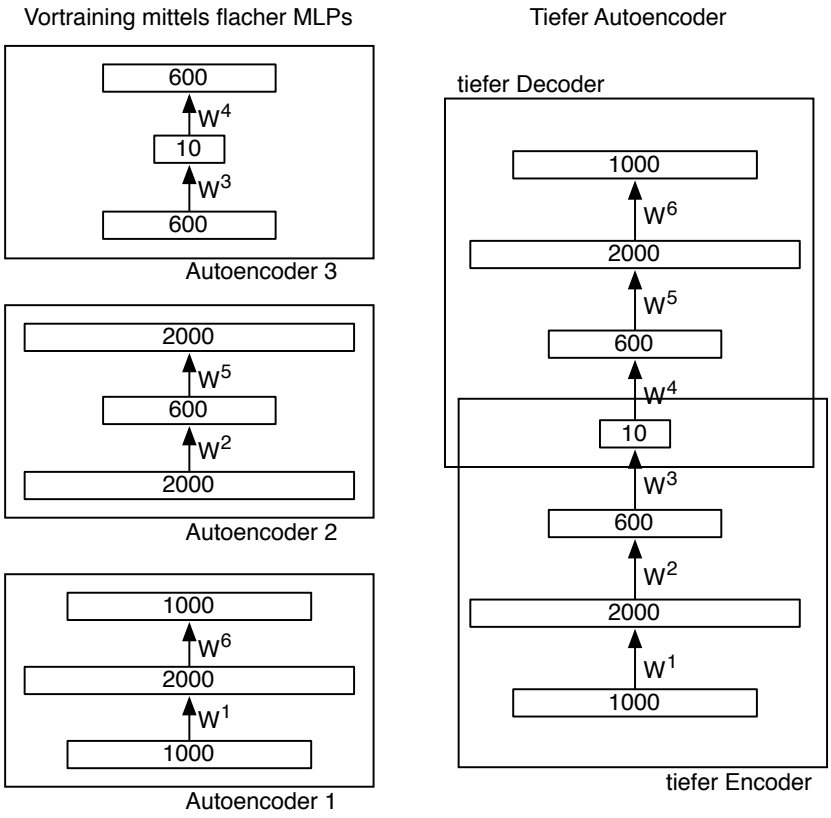


Abbildung 5.1: Vortraining mittels mehrerer flacher Autoencoder.

Im folgenden Abschnitt wird die praktische Implementierung dieses Algorithmus diskutiert, da die Effizienz der Umsetzung angesichts der Größe der trainierten Netze mit oftmals mehreren Millionen Verbindungsgewichten eine wichtige Rolle spielt.

5.2.2 Effiziente Implementierung

Tatsächlich kann das beschriebene Training der tiefen Netze mit minimalem Aufwand basierend auf bestehenden Bibliotheken zum Training von MLPs implementiert werden. Es müssen lediglich die entsprechenden Netze erzeugt und die Gewichte hin und her kopiert werden. Während die prinzipielle Umsetzung des schichtenweisen Vortrainings also auf konzeptionell recht einfache Weise geschehen kann, gilt es hingegen, der Effizienz besondere Aufmerksamkeit zu widmen. In interessanten Problemstellungen wächst die Zahl der Verbindungsgewichte allein schon aufgrund der oftmals hohen Eingabedimension schnell in die Zehntausende oder sogar Millionen. Um solche Problemstellungen, die unter Umständen auch noch das wiederholte Training eines Autoencoders erfordern, überhaupt angehen zu können, ist eine wesentlich höhere Verarbeitungsgeschwindigkeit

großer Netze erforderlich, als sie durch eine direkte Umsetzung der Gleichungen in eine rein sequentielle Verarbeitung zu erzielen wäre. So stößt zum Beispiel N++[1] beim Training tiefer Netze schnell an die Grenzen des Vertretbaren. Die von Salakhutdinov und Hinton beschriebene Trainingsprozedur eines Encoders mit 764-500-500-2000-30 Neuronen auf den 50 000 MNIST-Trainingsbeispielen [144] dauert mit N++ auf aktueller Hardware ca. einen Monat, für eine Anwendung in komplexeren Reinforcement Lernaufgaben und auf realen Systemen schlicht zu lange.

Aus diesen Gründen wurden einige Anstrengungen unternommen, die verschiedenen Möglichkeiten moderner Rechnerarchitekturen zur parallelen Verarbeitung für das Training auszunutzen. Da die naheliegende und einfach umzusetzende Idee, die Berechnung der Netzeingabe verschiedener Neuronen derselben Schicht parallel vorzunehmen, aufgrund des hohen Synchronisationsaufwands zwischen den dann recht kleinen Arbeitspaketen nicht zu Laufzeitverbesserungen führte, wurden schließlich folgende, wesentlich tiefgreifendere Änderungsmaßnahmen ergriffen, um die notwendige Beschleunigung von N++ zu erreichen:

- In N++ werden die einzelnen Gewichte bei dem jeweiligen Zielneuron in Form einer verketteten Liste gespeichert. Diese Methode bietet maximale Flexibilität bei der Verbindungsstruktur (jede Verbindung kann individuell angelegt werden) und ist für "kleine" Netze mit typischerweise deutlich unter hundert an einem Neuron eintreffenden oder abgehenden Verbindungen auch durchaus effizient. Aufgrund der Art der in N++ implementierten Speicherbelegung kann es aber zu einer Fragmentierung der zum selben Neuron führenden Gewichte kommen. Mit der Datenfragmentierung sind gerade bei den großen Datenmengen tiefer Netze Nachteile beim Caching verbunden, die zu deutlichen Geschwindigkeitseinbußen gegenüber einer besseren Speichernutzung führen können (siehe Tabelle 5.1). Außerdem steht diese Speicherorganisation der effizienten Anwendung von SIMD[2]-Architekturen wie Intels SSE/SSE2 entgegen, da für den Aufruf entsprechender Befehle die parallel zu verarbeitenden Daten von den meisten Bibliotheken in aufeinander folgenden Bereichen des Hauptspeichers erwartet werden.

  Aufgrund dieser Beobachtungen wurde die Speichernutzung von n++ komplett umorganisiert. Aktivierung, Netzeingabe, Gewichte und Gradienten eines Neurons werden nun nicht mehr zusammen in einer Struktur, sondern jeweils für eine ganze Netzwerkschicht in separaten Matrizen abgespeichert. Alle Gewichte zwischen Neuronen zweier aufeinander folgender Schichten werden so in einer Gewichtsmatrix W zusammenhängend im Speicher abgelegt. Die Gewichte zum Biasneuron werden, wenn benötigt, als Spalte 0 der Gewichtsmatrix eingefügt.

  So organisiert, können die notwendigen Operationen zum Propagieren der Aktivierungen und Gradienten mittels Operationen auf Matrizen und Vektoren implementiert werden, die sich durch Verwendung der auf moderne Prozessoren optimierten

[1] N++ ist eine am Lehrstuhl von Prof. Dr. Riedmiller entwickelte Implementierung von MLPs.
[2] Single Instruction Multiple Data

BLAS[1]-Bibliotheken [18, 87] dramatisch beschleunigen lassen (siehe Tabelle 5.1).

Tabelle 5.1: Vergleich der benötigten Rechenzeiten verschiedener Implementierungen des Skalarprodukts. Berechnet wurden 3000×1000 Skalarprodukte zwischen 1000-dimensionalen Vektoren, was dem Vorwärtspropagieren von 3000 Eingabevektoren durch 1 Mio Verbindungen zwischen zwei Schichten mit je 1000 Neuronen entspricht.

Methode	Zeit	Faktor
Verkettete Liste, fragmentiert (n++, worst case)	36.5s	1.0x
Verkettete Liste, zusammenhängend (n++, best case)	22.2s	1.6x
Vektoren, per Schleife	16.6s	2.2x
Vektoren, per CBLAS (Linux)	6.9s	5.3x
Vektoren, per CBLAS (Intel MKL, 32bit)	5.3s	6.9x

Für das Propagieren der Aktivierungen von einer Schicht mit n Neuronen zur nachfolgenden Schicht mit m Neuronen ist die Berechnung des Produkts

$$\begin{pmatrix} net_1 \\ net_2 \\ \vdots \\ net_m \end{pmatrix} = \begin{pmatrix} w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ w_{2,0} & w_{2,1} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,0} & w_{m,1} & \cdots & w_{m,n} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$$

nötig, das sich in BLAS durch eine Level 2 [34] Matrix-Vektor-Multiplikation umsetzen lässt. Das Rückwärtspropagieren der partiellen Ableitungen von den Neuronen der Schicht i zu denen der Schicht j lässt sich für ein Trainingspattern durch

$$\begin{pmatrix} \frac{\partial E_p}{\partial a_1} \\ \frac{\partial E_p}{\partial a_2} \\ \vdots \\ \frac{\partial E_p}{\partial a_n} \end{pmatrix} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix}^T \cdot \begin{pmatrix} \frac{\partial E_p}{\partial net_1} \\ \frac{\partial E_p}{\partial net_2} \\ \vdots \\ \frac{\partial E_p}{\partial net_m} \end{pmatrix}$$

ausdrücken, das Rückwärtspropagieren zu den Gewichten wird durch die Multiplikation

$$\begin{pmatrix} \frac{\partial E_p}{\partial w_{1,0}} & \frac{\partial E_p}{\partial w_{1,1}} & \cdots & \frac{\partial E_p}{\partial w_{1,n}} \\ \frac{\partial E_p}{\partial w_{2,0}} & \frac{\partial E_p}{\partial w_{2,1}} & \cdots & \frac{\partial E_p}{\partial w_{2,n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E_p}{\partial w_{m,0}} & \frac{\partial E_p}{\partial w_{m,1}} & \cdots & \frac{\partial E_p}{\partial w_{m,n}} \end{pmatrix}^T = \begin{pmatrix} 1 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial E_p}{\partial net_1} & \frac{\partial E_p}{\partial net_2} & \cdots & \frac{\partial E_p}{\partial net_m} \end{pmatrix}$$

vorgenommen, wobei die a_i die Aktivierungen der Neuronen der Schicht i bezeichnen und die partiellen Ableitungen $\frac{\partial E}{\partial net_i}$ sich auf die Neuronen der nachfolgenden

[1] Basic Linear Algebra Subprograms, http://www.netlib.org/blas, besucht im Feb. 2010

Schicht j beziehen. Die zweite Gleichung lässt sich als Level 3 [33] Matrix-Matrix-Multiplikation (Resultat ist eine von den Vektoren aufgespannte Matrix) und die erste Gleichung als Level 2 Matrix-Vektor-Multiplikation in BLAS umsetzen.

- Neben der mittels BLAS erzielten Beschleunigung, die u.a. von der gleichzeitigen Anwendung einer Operation auf mehrere in den Registern eines Prozessorkernes liegenden Werten herrührt, kann eine weitere Beschleunigung durch die Aufteilung der Arbeit auf verschiedene Prozessorkerne erzielt werden. Allerdings ist hierbei zu beachten, dass der zwischen notwendigen Synchronisationen durchführbare Arbeitsaufwand nicht zu gering ist und der Aufwand für die Erzeugung und den Start der Threads nicht den Geschwindigkeitsvorteil bei den Berechnungen dominiert.

  Die Beschränkung auf batch-mode Lernverfahren, bei denen zunächst alle Trainingsdaten bei unveränderten Gewichten durch das Netz propagiert und die Gradienten an den einzelnen Gewichten aufsummiert werden, erlaubt eine Parallelisierung in großen Verarbeitungsblöcken, die im Falle von online Aktualisierungen nicht möglich wäre: Die Summe $E = \sum_{p=1}^{k} E_p$ über alle k Trainingsbeispiele $(x_p; y_p) \in \mathcal{P}$ kann wegen der Assoziativität leicht in n Teilsummen aufgeteilt und auf n identischen Kopien des Netzes parallel berechnet werden:

$$\sum_{p=1}^{k} E_p = \underbrace{\sum_{p_1=0k/n+1}^{k/n} E_{p_1}}_{\text{1.Kopie}} + \underbrace{\sum_{p_2=1k/n+1}^{2k/n} E_{p_2}}_{\text{2.Kopie}} + \underbrace{\sum_{p_3=2k/n+1}^{3k/n} E_{p_3}}_{\text{3.Kopie}} + \cdots + \underbrace{\sum_{p_n=(n-1)k/n+1}^{k} E_{p_n}}_{n\text{-te Kopie}}$$

  Gleiches gilt für die Zerlegung der Summen der partiellen Ableitungen $\frac{\partial E}{\partial w_{j,i}} = \sum_{p=1}^{k} \frac{\partial E_p}{\partial w_{j,i}}$ nach den Gewichten. Die berechneten Teilsummen werden anschließend in ein gemeinsames Netz aufsummiert, so dass eine Gewichtsanpassung vorgenommen werden kann, die absolut identisch zur rein sequentiellen Berechnung ist. Mit den geänderten Gewichten werden dann die Gradienten der nächsten Epoche mit Hilfe der Netzkopien wieder in n Threads parallel berechnet. Der Vorteil dieses Ansatzes gegenüber einer Parallelisierung innerhalb des Netzes auf Ebene der Neuronen ist die Größe der Arbeitsblöcke. Es werden im Training mehrere Eingabevektoren durch das gesamte Netz geschoben, bevor erst vor einer Gewichtsänderung eine Synchronisation zwischen den Threads nötig ist.

In der Summe führen die gleichzeitige Verwendung mehrerer Register innerhalb eines Prozessorkerns und die Aufteilung der Arbeit auf verschiedene Prozessorkerne zu einer deutlichen Reduktion der Rechenzeit auf ca. $\frac{1}{28}$ der ursprünglich von n++ benötigten Zeit (siehe Tab. 5.2). Für das Training eines Autoencoders auf den MNIST-Datensatz benötigt die hier vorgestellte Implementierung ca. 22 Stunden, wenn der exakt gleiche Netzaufbau und die exakt gleiche Anzahl an Trainingsepochen verwendet wird, wie im von Hinton veröffentlichten Matlab-Sourcecode zu [61]. Hintons Matlab Implementierung benötigt auf demselben Rechner für das Experiment über 60 Stunden, wobei seine Implementierung die meiste Zeit bei der Durchführung der aufwändig zu berechnenden

Aktualisierungen mit konjugierten Gradienten in den 200 Epochen der Feintrainingsphase verliert[1].

Tabelle 5.2: Vergleich der Rechenzeiten zwischen alter und neuer Implementierung von n++. In der neuen Version kann n++ durch die Verwendung mehrerer Threads alle acht Prozessorkerne des Testsystems (Apple MacPro4,1 mit zwei 2,26 GHz Quad-Core Intel Xeon CPUs mit je 2 Threads pro Kern) auslasten, so dass die real benötigte Zeit (REAL) sinkt, auch wenn die Summe der auf allen Kernen benötigten Rechenzeit (USER) durch den Mehraufwand für die Synchronisation und durch andere Effekte (z.B. häufigeres Warten auf Speicherzugriff) ansteigt. Berechnet wurden zwei volle Trainingsepochen auf 10 000 MNIST-Trainingsbeispielen für ein dreischichtiges Netz mit 500 versteckten Neuronen und ca. 800 000 Verbindungsgewichten.

Methode	Threads	Zeit		Faktor
		real	user	real
n++	1	420.1s	419.7s	1.0x
n++ BLAS-Support	1	107.9s	107.7s	3.9x
n++ BLAS-Support, Multi-Threads	2	57.1s	113.7s	7.4x
n++ BLAS-Support, Multi-Threads	4	33.0s	138.4s	12.7x
n++ BLAS-Support, Multi-Threads	8	25.4s	165.4s	16.5x
n++ BLAS-Support, Multi-Threads	16	15.1s	228.6s	27.8x

Die gegenüber N++ erzielte siebenundzwanzigfache Beschleunigung geht über eine lediglich "komfortable" Aufwandsreduzierung weit hinaus. Erst durch die Parallelisierung und die deutliche Beschleunigung werden die später beschriebenen praktischen Lernversuche an realen Systemen überhaupt ermöglicht und können innerhalb von Stunden bzw. Tagen an Stelle von Wochen und Monaten durchgeführt werden. Es ist anzunehmen, dass die derzeitige Weiterentwicklung bei den Prozessoren und die Verwendung hoch paralleler Grafikprozessoren für das wissenschaftliche Rechnen [123, 153] in naher Zukunft zu weiteren, dramatischen Beschleunigungen beim Training parallelisierbarer neuronaler Netze führen und so die Grenzen lösbarer Lernaufgaben weiter verschieben werden. Es scheinen sich neben dem tiefen Lernen völlig neue, vormals geradezu utopische, Anwendungsfelder für große neuronale Netze und Datenmengen zu eröffnen.

5.2.3 Rezeptive Felder und Faltungskerne

Neben der schichtweisen Vollvernetzung sind in DFQ mit den rezeptiven Feldern und den Faltungskernen zwei weitere Verbindungsstrukturen implementiert, die insbesondere in den hier untersuchten Bildverarbeitungsproblemen zum Einsatz kommen können.

Rezeptive Felder Statt jedes Neuron einer Schicht mit allen Neuronen der Vorgängerschicht zu verbinden, kann auf den Netzschichten eine zu den Eingabebildern kor-

[1] Matlab nutzt sowohl SIMD als auch die Prozessorkerne für die Beschleunigung der von Hinton implementierten Matrixmultiplikationen aus.

respondierende, zweidimensionale Anordnung definiert werden, um dann nur Neuronen in einem räumlich begrenzten Einzugsbereich – dem "rezeptiven Feld" des Nachfolgeneurons – mit einem Neuron der Nachfolgeschicht zu verbinden. Diese Idee, nur eine begrenzte Anzahl benachbarter Pixel mit einem Neuron der Nachfolgeschicht zu verbinden, bezieht eine gewisse Motivation aus der Biologie – die Stäbchen und Zapfen im menschlichen Auge sind in ganz ähnlicher Weise verbunden und Formen räumlich begrenzte rezeptive Felder auf der Retina [62, 72] – und geht hinsichtlich der Verwendung in künstlichen neuronalen Netzen bis mindestens auf das Neocognitron [45] zurück.

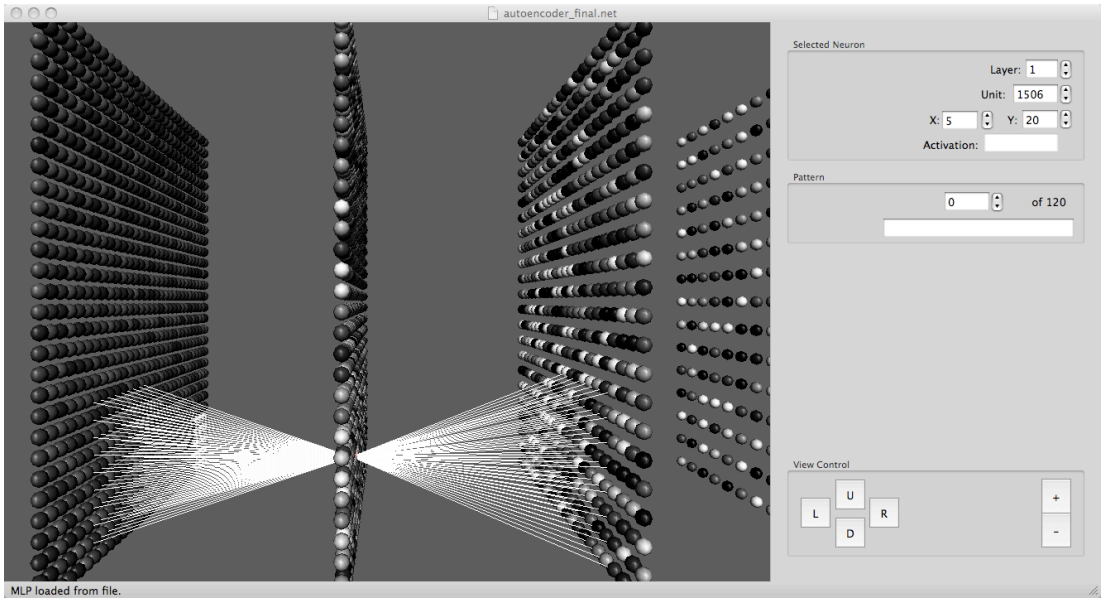


Abbildung 5.2: Teilansicht eines tiefen Netzes mit 9×9 Neuronen großen rezeptiven Feldern im zur Implementierung gehörenden OpenGL-NetViewer. Dargestellt sind die eingehenden und ausgehenden Verbindungen des Neurons an der x/y-Position $(6, 21)$ in der ersten versteckten Schicht.

Für die tiefen Autoencoder stellen die rezeptiven Felder eine Möglichkeit dar, einerseits die große Anzahl der Freiheitsgrade etwas zu reduzieren und andererseits den in den Daten nur implizit enthaltenen Zusammenhang zwischen den Dimensionen im Bild benachbarten Pixel auch explizit über die Netzstruktur auszudrücken. Offensichtlich ist die räumliche Nachbarschaft im Bild eine Eigenschaft, die auch im menschlichen Auge ausgenutzt wird. Gerade in den frühen Verarbeitungsschichten, wo sowohl in der biologischen als auch in der maschinellen Verarbeitung visueller Eindrücke die Hauptaufgabe zunächst die Detektion von lokalen Bildmerkmalen ist, erscheint der Verzicht auf die Bereitstellung globaler Bildinformationen und die Beschränkung auf die Weiterleitung lokaler Aktivierungen keine wesentliche Einschränkung zu sein.

In DFQ sind die rezeptiven Felder als quadratische Verbindungsbereiche mit einer ungeraden Kantenlänge (Anzahl der Neuronen) implementiert (siehe Abbildung 5.2). Im

Decoder werden die rezeptiven Felder gespiegelt, so dass sich eine achsensymmetrische Verbindungsstruktur mit der Kodierungsschicht als Mittelachse ergibt. Rezeptive Felder in den äußeren Schichten des Autoencoders können ohne Probleme mit vollvernetzten Schichten in den weiter innen liegenden Schichten kombiniert werden. Um die zweidimensionalen rezeptiven Felder sinnvoll in Bezug auf die Eingabebilder definieren zu können, ist es notwendig, die Neuronen in den betroffenen Schichten so in einer zweidimensionalen Struktur anzuordnen, dass ihre Position den Bildbereich widerspiegelt, von dem sie Eingänge erhalten. Diese neu eingeführte 2D-Struktur und die Korrespondenz zwischen den Schichten ist gut in Abbildung 5.2 zu erkennen.

Faltungskerne Anstelle in jedem der rezeptiven Felder eigene Gewichte für die Verbindungen zu verwenden, können die Gewichte zwischen den rezeptiven Feldern auch gemeinsam genutzt werden. Diese Idee der geteilten Gewichte geht auf LeCuns Convolutionary Neural Nets zurück [88] und ist schon seit einigen Jahren bekannt, wurde bisher aber noch nicht in Verbindung mit dem schichtenweisen Vortraining tiefer Autoencodernetze nach Hinton [61] verwendet.

Der beim Einsatz der geteilten Gewichte (engl. Weight-Sharing) entstehende Netzaufbau entspricht einer Faltungsoperation: Ein einzelner Kern wird quasi über die Neuronen der Vorgängerschicht geschoben und auf alle möglichen Positionen angewendet, um die Netzeingaben der Neuronen in der Nachfolgeschicht zu berechnen. Gerade im Falle eines im Bild wandernden Objekts, dessen Aussehen sich nicht ändert, ist diese Methode hilfreich. Das Training der Gewichte eines Kerns zur Detektion dieses Objekts geschieht ortsunabhängig; seine Gewichte werden mit allen Bildern effektiv trainiert, unabhängig davon, wo genau im Bild sich das Objekt gerade befindet.

Der Einsatz von Weight-Sharing kann so einen weiteren Beitrag zur Reduzierung der anzupassenden Verbindungsgewichte und zur weiteren Erleichterung des Trainings leisten. Einerseits werden durch die Trennung von Aussehen und Position weniger Daten benötigt, um das Aussehen des Objekts gut in den Gewichten zu erfassen, andererseits werden die Freiheitsgrade des Netzes über die Reduzierung der Anzahl der veränderbaren Gewichte weiter eingeschränkt. Auch die Robustheit gegenüber dem Rauschen einzelner Pixel kann durch die Anwendung des Weight-Sharings gesteigert werden.

Die Verwendung der Faltungskerne innerhalb der tiefen Autoencoder stellt eine große Reduzierung der Freiheitsgrade gegenüber der Verwendung von rezeptiven Feldern oder gar vollvernetzten Schichten dar. Werden die Kerne wie die rezeptiven Felder im Decoder gespiegelt, kann dies zu Problemen führen. In vielen Fällen ist eine gute Rekonstruktion der Daten im Decoder nicht möglich, da die durch die Faltung berechnete Operation mit den verfügbaren, wenigen Freiheitsgraden nicht umgekehrt werden kann. In der Praxis hat es sich daher ausgezahlt, nur im Encoder geteilte Gewichte einzusetzen, während der Decoder die Gewichte aber nicht teilt, sondern nur rezeptive Felder verwendet. Hier kommt ein Vorteil der eigenen Implementation zum Tragen, die es erlaubt, beliebige, asymmetrische Verbindungsstrukturen im Encoder und Decoder zu verwenden, solange nur die Anzahl der Neuronen in den Schichten gespiegelt wird.

5.2.4 Parameter und Modellauswahl

Im Einsatz von Autoencodern gibt es viele Parameter, die eingestellt werden können. Dies ist im Moment noch eine klare Schwäche des tiefen Lernens. Diese Problematik fängt mit der Auswahl eines geeigneten Modells an und hört mit der Einstellung der Trainingsparameter und Trainingsdauern auf. Während bei der Netztopologie die Größen der Eingabeschicht und der Kodierungsschicht quasi durch die Dimension der Bilder und der Kodierung, auf die man abzielt, festgelegt sind, sind die Anzahl der versteckten Schichten und die Schritte der Größenabsenkung und die damit verbundene Anzahl der Neuronen in den einzelnen Schichten frei wählbar, ohne dass vorab völlig klar wäre, welche Werte hier zum Erfolg führen. In einigen Experimenten hat es sich sogar als lohnenswert erwiesen, die Dimension in der ersten versteckten Schicht zunächst aufzublähen, und erst dann in Richtung der angestrebten Merkmalsraumdimension abzusenken [61]. Die Unsicherheit bei der Parameterauswahl geht derzeit so weit, dass von einer Publikation zur anderen – und auch selbst innerhalb derselben Publikation – oftmals völlig andere Architekturen in den einzelnen Experimenten und Auswertungen – auch für ein und denselben Datensatz – verwendet werden.

Durch die im Rahmen dieser Arbeit in die Autoencoder eingeführten rezeptiven Felder und Faltungskerne kommen mit der Größe der Einzugsbereiche, der Anzahl der in einer Schicht verwendeten unterschiedlichen Kerne, der Auswahl der Schichten, die mit Kernen, rezeptiven Feldern oder voller Vernetzung arbeiten und der Verschaltung der Ergebnisse einzelner Kerne in darauf folgenden Schichten – hier sind fast beliebig komplizierte Stukturen denkbar [88, 124] – sogar noch weitere Parameter hinzu. Kleinigkeiten sind dabei schon die Entscheidung, ob auch Biasgewichte geteilt werden (wäre enger an der Faltungsoperation) und ob die Kerne auch bis in die Ecken, über den Rand des Bildes hinaus, angewendet werden.

Aufgrund der hohen Laufzeiten ist es zudem oft nicht möglich, viele verschiedene Modelle auszutesten, geschweige denn eine systematische Evaluation über alle möglichen Parameter durchzuführen und das wirklich optimale Modell zu bestimmen. Tatsächlich ist die Auswahl der Topologie und Verbindungsstruktur daher immer noch mehr eine Kunst denn eine Wissenschaft und benötigt vor allem Erfahrung. Immerhin wurde inzwischen eine rechenintensive, systematische Evaluation einiger der wichtigen Topologieparameter in schichtenweise vollvernetzten Netzen auf dem MNIST-Datensatz durchgeführt [38], so dass man einige Anhaltspunkte für die Wahl der Parameter zur Verfügung hat – auch wenn diese Ergebnisse natürlich nur für den untersuchten MNIST-Datensatz zuverlässig sind und auf anderen Datensätzen stark abweichen können.

In den ersten eigenen praktischen Versuchen hat sich die Abhängigkeit des Resultats von zur Aufgabe passenden Modellparametern voll bestätigt. So war die Übertragung einer auf dem MNIST-Datensatz erfolgreichen Architektur auf ein Grid-World-Problem (Beschreibung des Lernproblems in Abschnitt 5.3.1) nicht erfolgreich. Das schichtenweise vollvernetzte Netz konnte auf einem Satz von 775 leicht verrauschten Bildern nicht auf die Erzeugung guter Rekonstruktionen trainiert werden und verharrte bei der Rekonstruktion des Durchschnittsbildes für alle Eingaben. Über die Beobachtung der Fehler im Vortraining der einzelnen Schichten konnte in zahlreichen weiteren Versuchen

schließlich eine Netzarchitektur gefunden werden, die im Grid-World-Problem gute Resultate liefert.

Aufgrund dieser Ergebnisse wurden einige vorläufige Lernversuche mit unterschiedlichen Parametern und Datensätzen unternommen, um Erfahrungen zu sammeln und die Freiheitsgrade bei der Parameterauswahl etwas einzuschränken. Dabei wurde der Anspruch einer optimalen Lösung jedes einzelnen Lernproblems – diese ist von einer individuellen Einstellung der Parameter abhängig – zugunsten einer Architektur aufgegeben, die ohne großes Finetuning auf möglichst vielen Problemen und Datensätzen ein gutes – aber nicht unbedingt optimales – Ergebnis liefert. Das Resultat ist eine in Teilen generische Architektur, mit der sich auf vielen der hier behandelten Navigationsprobleme gute Merkmalsräume erzeugen lassen. Eine zuverlässige, robuste Netztopologie ist eine der Voraussetzungen, damit später die automatische Erzeugung von hilfreichen Merkmalsräumen in DFQ gelingen kann.

In der Folge seien kurz die in der Netzarchitektur[1] verwendeten Parameter und die gefundenen Standardeinstellungen (im Vorgriff auf Ergebnisse aus der Evaluation und aus Kapitel 7) aufgeführt:

Netztopologie	
w, h	Größe der Eingabeschicht – vorgegeben durch Bildgröße
r_b	Verkleinerungsfaktor, $\|N_{k+1}\| = \|N_k\|/r_b$ – üblicherweise $r_b = 2$
r_1	Verkleinerungsfaktor der 1. versteckten Schicht – üblicherweise $r_1 = 1$
n_t	Anzahl der Neuronen in der Kodierungsschicht
n_{t-1}	Minimalanzahl der Neuronen direkt vor der Kodierungsschicht – üblicherweise $n_{t-1} = r_b n_t$

Verbindungsstruktur	
d_k	Dimension des rezeptiven Feldes ($k \times k$ Neuronen) – ungerade Zahl
weight-sharing	Verwendung geteilter Gewichte (Faltungskerne) – boolean
bias-sharing	Verwendung geteilter Biasgewichte – boolean, üblicherweise 'nein'
overlapping	Über den Rand ragende Felder – boolean, üblicherweise 'ja'
n_k	Anzahl der Felder je Schicht – üblicherweise 1
l_k	Anzahl der mit Feldern verbundenen Schichten – üblicherweise zwischen 2 und 4

Lernparameter	
σ_0	Standardabweichung bei der Initialisierung der Gewichte der *flachen* Autoencoder – üblicherweise 0.5
Δ_0	Initiale Änderung, RProp – üblicherweise $\frac{1}{10}\Delta_{\max}$
$\Delta_{\max}$	Maximale Änderung, Rprop – üblicherweise 0.1
λ	Weight-decay – üblicherweise $\frac{1}{1000}\Delta_{\max}$

Von dieser Vielzahl der Parameter werden in den praktischen Lernversuchen meist nur sehr wenige Parameter individuell gewählt und ansonsten die bewährten Standardeinstellungen verwendet. Da die Größe der Eingabeschicht $w \times h$ und die Dimension n_t des

[1] Es gibt in der Implementierung einen Netzgenerator, der mit diesen Informationen gefüttert wird und dann einen passenden Autoencoder erzeugt.

angestrebten Merkmalsraums von der Aufgabenstellung vorgegeben werden, begrenzt sich die Auswahl zumeist auf die Entscheidung für oder gegen den Einsatz der Faltungskerne (Weight Sharing), die Größe d_k des Einzugsbereiches der verbundenen Felder und die Verkleinerungsrate r_b. Durch die Wahl einer einheitlichen Verkleinerungsrate für alle Schichten werden in dieser Architektur zumeist wesentlich mehr versteckte Schichten als in den von Hinton handoptimierten Netzen verwendet. Dies kann zu etwas schlechteren Ergebnissen als mit kleineren Netzen führen [38, 83], gestaltet sich in der Praxis aber wesentlich unproblematischer, da nicht nach der Einstellung jeder einzelnen Schichtgröße überprüft werden muss, ob die Informationen noch hindurchgeführt werden können oder schon verloren gehen.

Aufgepasst werden muss bei einer Verkleinerung des Parameters n_{t-1} gegenüber dem Standardwert, denn in den durchgeführten Versuchen hat sich gezeigt, dass eine zu starke Verkleinerung der Schichten vor der Kodierungsschicht oft zu deutlich schlechteren Ergebnissen führt. Kritisch ist auch der Parameter l_k, der die Anzahl der versteckten Schichten direkt nach der Eingabeschicht angibt, die mit rezeptiven Feldern bzw. Kernen verbunden und nicht vollvernetzt werden. In der Praxis können meist mehr Schichten mit rezeptiven Feldern ($l_k \leq 4$ unproblematisch) als Schichten mit Kernen (in allen Experimenten hat $l_k = 2$ funktioniert) verwendet werden. Werden zu viele dieser Schichten verwendet, kommt es zu keinem guten Trainingsergebnis mehr und selbst die Trainingsdaten werden vom Autoencoder nur auf das Durchschnittsbild abgebildet.

Ein positiver Aspekt ist die Einstellung der Lernparameter. Hier zahlt sich der Einsatz von RProp aus, das ganz allgemein als sehr robust gegenüber der Parameterauswahl gilt. Die für die Lernparameter aufgeführten Werte $\Delta_{\max}$, Δ_0, die Regularisierung λ und die zufällige Initialisierung der Gewichte σ_0 führten in allen Experimenten zum Erfolg, wobei die Angabe von $\Delta_{\max}$ sich auf den Wert im schichtenweisen Vortraining bezieht und im Feintraining immer auf ein Zehntel dieses Werts abgesenkt wird.

5.3 Empirische Evaluation

Die innerhalb von DFQ eingesetzten Autoencoder sollen dazu dienen, aus den visuellen Beobachtungen automatisch einen niedrigdimensionalen Merkmalsraum zu konstruieren, auf dem erfolgreich Strategien erlernt werden können. An dieser Stelle sollen nun die von den tiefen Autoencodern erzeugten Merkmalsräume auf eine grundsätzliche Eignung hin untersucht werden. Die beiden Aspekte, die dabei im Vordergrund stehen – die im Reinforcement Lernen naturgemäß ungleichmäßige Verteilung der Daten und die Differenzierungs- und Generalisierungmöglichkeiten im erzeugten Merkmalsraum – wurden bereits in der Einleitung diskutiert. Besonderes Augenmerk wird auf die dort genannten vier Kriterien gelegt: Keine Verdeckungen, Robustheit gegenüber Rauschen, Nachbarschaft ähnlicher Zustände, Erhaltung der Topologie.

Bei der Untersuchung der Merkmalsräume haben sich im Bereich des tiefen Lernens im Wesentlichen die visuelle Begutachtung der erzeugten Merkmalsräume und die Durchführung von Klassifikationsexperimenten auf Basis der erzeugten Merkmalsvektoren als Analyseverfahren etabliert [61, 84]. Dazu werden insbesondere die im Training

nicht verwendeten, zugrunde liegenden Klassenzugehörigkeiten bzw. die bilderzeugenden Parameter zur Markierung der Merkmalsvektoren in den Visualisierungen und als Zielwerte der Klassifikation verwendet. Diesem Paradigma folgt auch ein Großteil der hier vorgenommenen Untersuchungen.

Der Trainingsalgorithmus wird zunächst auf einen Satz von künstlichen Bildern erprobt, wie sie in einem typischen RL-Benchmarkproblem vorkommen könnten. Die Verwendung von synthetischen Bildern, bevor das Verfahren im Abschnitt 5.3.7 auf realen Bildern erprobt wird, hat den Vorteil, dass die Parameter der Bildformation wie die zugrunde liegenden Systemzustände in den Experimenten unter völliger Kontrolle stehen, Umgebungseinflüsse und andere Fehlerquellen ausgeschlossen und die Vergleichsauswertungen so deutlich erleichtert werden können.

Tabelle 5.3: Übersicht zur Evaluation. Die Spalten Synth. und Real enthalten den Seitenverweis auf das jeweilige Experiment mit synthetischen Bildern und – sofern durchgeführt – mit realen Bildern. Die darauf folgenden Spalten enthalten eine Markierung der Relevanz des Experiments (○ → gewisse Relevanz, ● → hohe Relevanz) für die Kriterien Zustandsidentifizierende Eigenschaft (K1), Robustheit (K2), Nachbarschaft ähnlicher Zustände (K3), die Übertragung der Topologie (K4) und das Training auf kleinen, ungleichmäßig verteilten Datensätzen (DS).

Versuch / Auswertung	Datensatz		Relevanz				
	Synth	Real	K1	K2	K3	K4	DS
Rekonstruktion	S. 108	S. 122	●	●	○		
Visuelle Analyse Merkmalsraum	S. 108	S. 123	○	●	●		
Entwicklung des Merkmalsraums	S. 109	S. 124				○	
Klassifikation im Merkmalsraum	S. 109		●	●	○		
Visuelle Analyse Topologie	S. 110	S. 123			○	●	
Vergleich zu linearer Methode	S. 112						
Verbesserung Merkmalsraum	S. 115		○		○	●	○
Faltung vs. rezeptive Felder	S. 120		○		○		●
Kleine Trainingsmengen	S. 120	S.125					●
Vergleich Kerngrössen	S. 120		○		○		○
Künstliches Rauschen	S. 121			●			

In Tabelle 5.3 sind der Übersicht halber alle durchgeführten Experimente aufgelistet und hinsichtlich ihrer Relevanz für die Kriterien 1 bis 4 und das Training auf kleinen, ungleichmäßig verteilten Datenmengen markiert. In Abschnitt 5.3.1 wird zunächst die verwendete kontinuierliche Grid-World mit synthetischer Bildformation beschrieben, bevor in Abschnitt 5.3.2 die Ergebnisse einiger grundsätzlicher Auswertungen dargestellt werden. Anschließend werden einige Methoden besprochen, um die Gewichte der Autoencoder in der Anwendung weiter zu optimieren. Mit dem Einsatz von Faltungskernen werden die Probleme auf kleinen Datensätzen bekämpft. Die erzielten Ergebnisse werden dann auf einem Satz realer Bilder in Abschnitt 5.3.7 verifiziert. Am Ende jedes Teilabschnitts dieser Evalution folgt direkt eine kurze Diskussion der in dem jeweiligen

Experiment erzielten Ergebnisse.

5.3.1 Kontinuierliche Grid-World

Für die Evaluation wird ein klassisches Benchmark-Problem verwendet – genau genommen eine kontinuierliche Variante der Grid-World [142] mit Mauern – von dem synthetische Bilder mit 30×30 Pixeln in Aufsicht erzeugt werden (siehe Abbildung 5.3). In diesem Problem kann jedes Pixel einen reellwertigen Grauwert zwischen 0 und 1 annehmen. Die 900-dimensionalen Bilder $o \in [0,1]^{900}$ zeigen einen rechteckigen, hellen Agenten, der sich durch ein $6m \times 6m$ großes, dunkles Labyrinth bewegt. Einige im Bild hell markierte Felder (Wände) können nicht betreten werden. Eine graue Fläche markiert den zu erreichenden Zielbereich, der sich in einem durch Wände abgetrennten Zielraum befindet und nur durch einen engen Flaschenhals erreicht werden kann. Die Kantenlängen der Wandelemente, der Zielfläche und des Agenten betragen jeweils $1m$. Der Mittelpunkt $s = (x, y)$ des Agenten kann sich an jeder beliebigen, kontinuierlichen Position $s \in [0, 6)^2$ innerhalb des Labyrinths befinden, die nicht durch eine Wand belegt ist. Abschließend werden die vom System synthetisierten Bilder mit additivem Zufallsrauschen $\mathcal{N}(0, \sigma)$ versehen. Die Standardabweichung beträgt in den meisten Versuchen $\sigma = 0.1$, wobei in einem Versuch auch variiert wird.

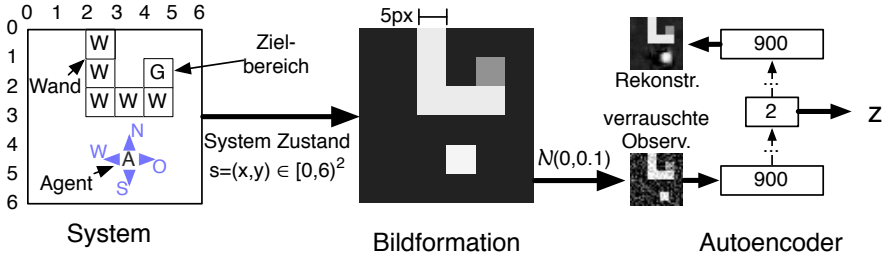


Abbildung 5.3: Darstellung des Versuchsaufbaus im kontinuierlichen Grid-World-Problem mit künstlicher Bildformation. Der Mittelpunkt des Agenten kann sich in jedem der 775 nicht durch eine Wand belegten Pixel befinden.

Hier spielen die Aktionen und die Kosten anders als später in Kapitel 7 noch keine Rolle, da für die Auswertungen der Merkmalsräume in diesem Kapitel zunächst nur die Bilder benötigt werden. Ziel ist es, mit Hilfe der Autoencoder auf diesen hochdimensionalen Bilddaten einen zweidimensionalen Merkmalsraum zu konstruieren, in der Hoffnung, dass sich die Positionsinformation s des Agenten in diesen Merkmalen z möglichst gut widerspiegelt. Dieses kontinuierliche Grid-World-Problem kennzeichnen dabei folgende Besonderheiten:

- Trotz der reellwertigen Position des Agenten ist die Anzahl der möglichen Bilder – aufgrund der diskreten Natur der Pixel – zunächst auf maximal $900 - 125 = 775$ (Gesamtanzahl der Pixel abzüglich Wände) begrenzt; Aliasing (hier im Sinne der Bedeutung in der Computergrafik) aufgrund nur teilweise durch den Agenten

abgedeckter Pixel wird in der synthetischen Bildformation nicht berücksichtigt und tritt erst in den Versuchen auf realen Bildern auf.

- Die "Tür" zum Zielraum stellt in den späteren RL-Experimenten mit Exploration eine interessante Herausforderung für DFQ dar. Eine reine Zufallsstrategie wird den Agenten nicht häufig durch diesen Flaschenhals führen, so dass es nur wenige Observationen von ihm geben wird. Dennoch muss dieser Flaschenhals zweifelsfrei im Mekrmkalsraum identifiziert werden können, denn nur so können niedrige Kosten aus dem Zielraum heraus propagiert werden.

- Das beim Einsatz realer Kameras nicht zu vermeidende Pixelrauschen wird durch das additive Zufallsrauschen simuliert – und nicht wie bei [40, 51, 69] einfach ignoriert.

- Prinzipiell sorgt allein das additive Bildrauschen dafür, dass zu jeder der 775 möglichen Bildpositionen des Agenten unendlich viele Observationen erzeugt werden können. Aufgrund der hohen Dimension und der 900 beim Verrauschen gezogenen Zufallszahlen ist es extrem unwahrscheinlich, dass jemals irgendeine Observation in einem Experiment zweimal erzeugt wird. Es muss also zwangsläufig in der Test- bzw. Anwendungsphase generalisiert werden.

- Dieses Navigationsproblem hat den Vorteil gegenüber komplexeren Problemen, dass es eine direkte, nachprüfbare Entsprechung zwischen Observation o, Pixel-Koordinaten des Agenten im Bild und dem internen, nicht beobachtbaren Systemzustand s gibt, die sich in den anschließenden Visualisierungen ausnutzen lässt und die Auswertung und Beurteilung ungemein vereinfacht.

In den Experimenten werden unterschiedlich große Datensätze von der synthetischen Bildformation in Zufallsexperimenten gezogen. Allen Experimenten ist gemein, dass nur die Hälfte der Bilder für das Training verwendet wird, während die andere Hälfte zum Testen der Generalisierungsleistung dient.

5.3.2 Grundlegende Ergebnisse

In diesem Abschnitt wird zunächst über einen grundsätzlichen Lernversuch berichtet und es werden einige Auswertungen eingeführt, die auch in den weiteren Untersuchungen eine Rolle spielen. Für den Versuch werden 6 200 über die möglichen Agentenpositionen gleichmäßig verteilte Bilder gezogen. 3 100 dieser Bilder werden im Training, der Rest in den Auswertungen verwendet. Als Modell hat sich ein Autoencoder mit 21 Schichten und rezeptiven Feldern mit je 9×9 Neuronen zwischen den vier äußersten versteckten Schichten bewährt in einigen verkürzten Testläufen auf einem Teil der Trainingsdaten bewährt. Der von diesem Autoencoder abgeleitete Encoder hat 900-900-484-225-121-113-57-29-15-8-2 Neuronen in seinen 11 Schichten und ca. 160 000 Verbindungsgewichte.

Rekonstruktion Der Autoencoder erzielt nach Abschluss des Vortrainings einen durchschnittlichen Rekonstruktionsfehler (siehe Abschnitt 2.1.3) von 17 pro Testbild und verbessert sich in der Finetuningphase auf 7.3. In Abbildung 5.4 sind einige Bilder der Testmenge und deren vom Autoencoder erzeugte Rekonstruktionen nach Abschluss des Trainings zu sehen. Eine exakte Rekonstruktion des Agenten im schmalen Durchgang zum Zielraum fällt dem Autoencoder auch auf weiteren Testbildern sichtbar schwerer als an den anderen Stellen in der Grid-World.

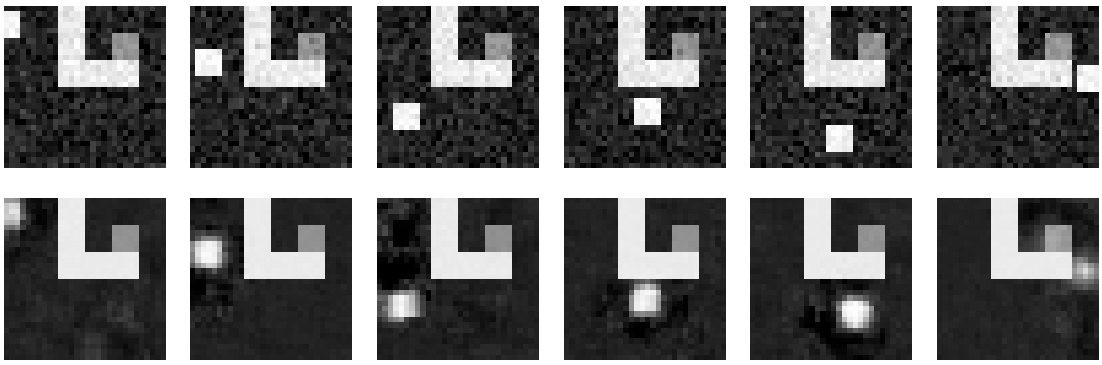


Abbildung 5.4: Einige Testbilder des Grid-World-Problems (obere Reihe) und vom Autoencoder erzeugte Rekonstruktionen (untere Reihe).

Ein Autoencoder mit identisch aufgebauter Netztopologie mit untereinander vollvernetzten Schichten und weit mehr als 1 000 000 Verbindungsgewichten allein im Encoder, wie sie von Hinton erfolgreich auf die MNIST-Daten angewendet wurde [61], erzielt einen deutlich schlechteren Rekonstruktionsfehler von 9.8 auf diesem Datensatz.

Visuelle Analyse des Merkmalsraums Interessanter als die eigentliche Rekonstruktionsgüte ist die Qualität der gefundenen Merkmale. In den Graphen in Abbildung 5.5 ist die Lage der Merkmalsvektoren aller Testbilder im zweidimensionalen Merkmalsraum zu sehen, der von den Ausgabeneuronen des Encodernetzes aufgespannt wird. Zu Visualisierungszwecken wird über die Grid-world ein Gitter von 6×6 Zellen gelegt. Alle Observationen, die den Mittelpunkt des Agenten in derselben Zelle zeigen, sind in der Abbildung mit dem gleichen farbigen Symbol markiert. Es sei noch einmal ausdrücklich darauf hingewiesen, dass diese Klassenzugehörigkeit zur gleichen Zelle während des Trainings nicht zur Verfügung steht und erst nachträglich aus Gründen der Visualisierung hinzugezogen wird.

Die durch das Gitter entstehenden Zellen im Labyrinth haben einen Zusammenhang mit den später in Kapitel 7 zu lernenden Wertfunktionen und Strategien: Aufgrund der verwendeten Schrittlängen und der diskreten Zeitschritte gibt es im Zustandsraum zusammenhängende Bereiche, aus denen der Agent von jeder Position (innerhalb eines Bereichs) gleich viele Schritte benötigt, um das Ziel auf dem kürzesten Weg zu erreichen. Die Bereiche haben die gleichen Ausmaße wie die Zielregion und decken sich mit den durch das Gitter erzeugten Zellen.

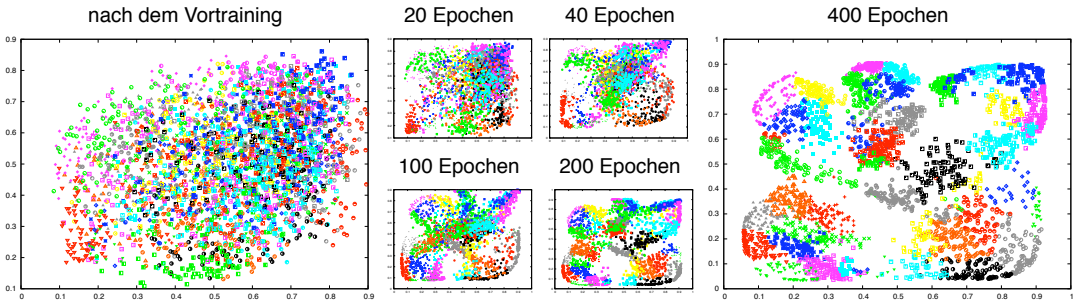


Abbildung 5.5: Entwicklung des Merkmalsraums während des Feintrainings des Autoencoders. Merkmalsvektoren von Bildern, die den Agenten in ähnlicher Position zeigen, sind mit dem gleichen farblichen Symbol markiert (siehe Text).

Der tiefe Autoencoder ist in diesem Fall eindeutig in der Lage, die relevanten Informationen (Position des weißen Agenten im Bild) automatisch aus dem Bild zu extrahieren. Trotz Pixelrauschen und leicht variierender Position befinden sich Merkmalsvektoren der Bilder des gleichen internen Systemzustands in unmittelbarer Nähe (Kriterium 2 und 3) und können gut von den Merkmalsvektoren anderer Positionen getrennt werden (Kriterium 1).

Entwicklung des Merkmalsraums Aufgrund der sehr kleinen bei der Initialisierung verwendeten Gewichte bildet der Autoencoder vor dem schichtenweisen Vortraining alle Trainingsbilder auf nahezu gleiche Merkmalsvektoren mit sehr geringer Varianz ($\sigma^2 < 0.01$) nahe $(0.5, 0.5)$ ab. In der Abbildung 5.5 ist die Entwicklung der Einbettung der Bilder im Merkmalsraum über den Verlauf des Feintrainings dargestellt. Es sind mehrere Dinge zu erkennen:

- Die Prozedur des Vortrainings bewirkt ein Auseinderziehen der Daten und führt zu einer guten Streuung im Merkmalsraum, noch nicht aber zu einer sinnvollen Anordung mit guter Separierung der Vertreter unterschiedlicher Klassen.
- Während des Finetunings wandern die Merkmalsvektoren der Testbilder durch den Merkmalsraum und bilden immer besser sichtbare Häufungen.
- Nach 400 Epochen ist eine klare Ordnung zu erkennen. Die Kodierungen von Bildern der gleichen Zelle bilden klar sichtbare, eng zusammenliegende Cluster, die sich kaum noch überlagern. Es sind insgesamt nur noch sehr wenige, einzelne Ausreißer zu entdecken, die zwischen den Kodierungen einer fremden Klasse liegen.

Klassifikation der Merkmalsvektoren Eine überwachte Lernaufgabe auf Basis der Merkmalsvektoren dient dazu, den Nutzen des gefundenen Merkmalsraums in Hinblick auf daran anknüpfende Lernaufgaben zu untersuchen. Der Ausgabevektor des Encoders

z für einen Eingabevektor o wird in diesem Fall als Eingabe für ein weiteres MLP verwendet, das dann auf die Approximation einer vorgegebenen Zielfunktion – zum Beispiel eine Klassifikation wie in [61] – trainiert wird. Im konkreten Fall soll eine quasi "inverse" Abbildung erlernt werden, die den Merkmalsvektoren als Klassenlabel die zugehörige Zelle des 6×6 Gitters zuweist. Trainiert wird dabei auf die x/y-Koordinaten des Mittelpunktes (geeignet normalisiert) der entsprechenden Zelle. Neben der Messung des durchschnittlichen quadratischen Fehlers zwischen Netzausgabe o und Zielwert y (engl. mean squared error, MSE) mit

$$\text{MSE} = \sum_{(x;y)\in\mathcal{P}} \sum_{i} (o_i - y_i)^2 \Big/ |\mathcal{P}| \ .$$

kann dann auch eine Klassifikationsrate (CR) bestimmt werden, indem die Merkmalsvektoren als richtig klassifiziert angesehen werden, die näher an den Koordinaten der richtigen Zelle liegen, als an den Koordinaten jeder anderen Zelle. Diese Methode gewinnt zusätzliche Aussagekraft dadurch, dass die Zellen des Gitters genau den Bereichen mit gleichen erwarteten Kosten unter der optimalen Strategie (siehe vorhergehender Abschnitt) entsprechen. Die Zellen stellen damit als Klassifikationsziel einen guten Test dafür dar, ob die Mekrmalsvektoren später eine gute Basis zur Approximation der optimalen Kosten bilden.

Da die internen Systemzustände dem Autoencoder während des Trainings nicht bekannt sind, ist diese Aufgabe prinzipiell nicht leichter zu erlernen als jede andere Regressions- oder Klassifikationsaufgabe, die Ergebnisse sind aber besonders gut zu interpretieren. Diese Aufgabe lässt sich zudem nur lösen, wenn die Position als wesentliche Information von den Autoencodern aus dem Bild extrahiert und in den Merkmalsvektoren gut repräsentiert wird. Für einen geringen Trainingsfehler und eine gute Generalisierungsleistung spielen die unter den Kritieren 1 bis 3 genannten Aspekte eine zentrale Rolle, so dass der Generalisierungsfehler als wichtiger Indikator für die Güte der erzeugten Merkmalsräume und die Stabilität der Merkmalsvektoren gegenüber Rauschen herangezogen werden kann.

Zunächst wird in einer Versuchsreihe eine Netzarchitektur gesucht, die auf einer von den Trainingspattern (3 100 Merkmalsvektoren der Trainingsbilder) abgetrennten Validierungsmenge einen möglichst kleinen Testfehler erzielt. In Frage kommen Netze mit einer oder zwei versteckten Schichten, wobei die Anzahl der Neuronen je versteckter Schicht in 5er Schritten von 5 bis 150 getestet wird. Eine weitere Einschränkung bei den Topologien ist, dass bei den zweischichtigen Netzen nur gleich große versteckten Schichten ausprobiert werden.

Aufgrund der vielen nichtlinearen Faltungen und Deformationen im Merkmalsraum ist es nicht erstaunlich, dass relativ viele versteckte Neuronen benötigt werden, um ein gutes Trainingsergebnis zu erzielen. Ein Netz mit der in den Versuchen besten Topologie mit je 25 Neuronen in den Schichten erreicht nach Abschluss des Trainings einen MSE von 0.035 auf den Testbildern und klassifiziert diese zu 80.10% korrekt.

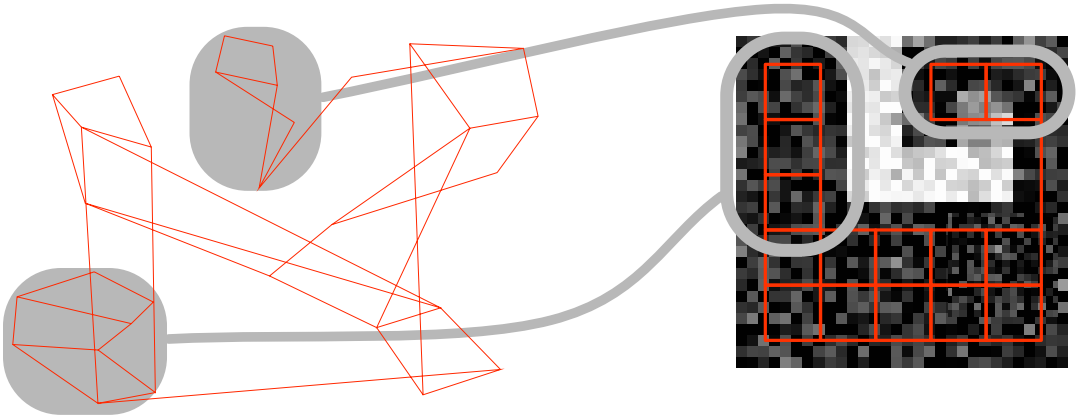


Abbildung 5.6: Nachbarschaftsbeziehungen im Merkmalsraum (links) des fertig trainierten Autoencoders. Die Knoten des Graphen repräsentieren die Position einiger Bilder im Merkmalsraum, wobei jedes Bild den Agenten im Zentrum einer der Gitterzellen zeigt (siehe Text). Knoten benachbarter Zellen sind mit einer roten Kante verbunden.

Visuelle Analyse der Topologie Hinsichtlich des vierten Kriteriums, der Widerspiegelung der ursprünglichen Topologie im Merkmalsraum, sind die Resultate uneinheitlich, aber insgesamt vielversprechend. In Abbildung 5.6 ist die Abbildung der ursprünglichen Topologie (Verbindungsgitter zwischen den Zellmittelpunkten des schon erwähnten 6×6 Gitters) im Merkmalsraum zu sehen. Lokal wird die Topologie an einigen Stellen im Merkmalsraum sehr gut erfasst. Zum Beispiel sind im Verbindungsgraphen unten links 8 Knoten zu sehen, die entsprechend ihrer Nachbarschaft im internen Zustandsraum richtig angeordnet werden, so dass es zu keinen Überschneidungen zwischen den sie verbindenden Kanten kommt. Diese Knoten entsprechen den 8 Vertices, die sich im Labyrinth links neben der Wand, außerhalb des Zielraums befinden. Auch an anderen Stellen sind lokal korrekt abgebildete Nachbarschaften zu erkennen. Global betrachtet ist das Ergebnis aber schlechter. Es sind einige Deformationen und Faltungen der Topologie zu beobachten; benachbarte Zustände sind auseinander gerissen und befinden sich an entgegengesetzten Enden des Merkmalsraums. In Abschnitt 5.3.4 wird untersucht, wie die Topologie unter Hinzuziehung von zusätzlichen Informationen noch weiter verbessert werden kann.

Diskussion Die Ergebnisse dieser Versuche zeigen, dass die tiefen Autoencoder prinzipiell in der Lage sind, die relevanten Informationen aus den von der Grid-World gewonnenen Bildern zu extrahieren und in nur zwei Dimensionen zu kodieren. Im erzeugten Merkmalsraum sind lokale Häufungen (Kriterium 3) mit klaren Grenzen (Kriterium 1 und 2) im Zustandsraum benachbarter Bilder zu erkennen. Auf Basis der Merkmalsvektoren lassen sich gute Klassifikationsergebnisse erzielen (Kriterium 1 und 2). In manchen Bereichen des Merkmalsraums ähnelt die Topologie derjenigen im ursprünglichen Zustandsraum.

Diese Ergebnisse erscheinen umso eindrucksvoller, wenn man bedenkt, dass in diesen Experimenten 900-dimensionale Bilder ohne weitere Informationen in ein hochkomplexes neuronales Netz eingespeist wurden, dessen mehrere Hundertausend Verbindungen auf nur wenigen Tausend Trainingsbeispielen so trainiert wurden, dass am Ende die Koordinaten des Mittelpunktes eines sich im Bild bewegenden Objektes für 4 von 5 verrauschten Testbildern korrekt ausgegeben werden – ein Ergebnis, dass noch vor nicht allzu langer Zeit alleine aufgrund des untrainierbar großen Netzes als völlig utopisch galt. Schaut man sich darüber hinaus an, wie sich der Klassifikationsfehler von 19.90% auf die Entfernungen vom korrekten Feld verteilt, wird deutlich, dass das Netz bei mehr als 90% der falsch klassifizierten Bilder nur um ein Feld daneben liegt, was den Fehler angesichts des Bildrauschens deutlich relativiert.

5.3.3 Vergleich mit der Hauptkomponentenanalyse

Um den Vorteil zu unterstreichen, den der Einsatz nichtlinearer, tiefer Encoder in den Navigationsaufgaben gegenüber dem Einsatz linearer Methoden bietet, wirde die gleiche Klassifikationsaufgabe auf Basis einer linearen Kodierung durchgeführt. Verwendet werden hierzu die von einer Hauptkomponentenanalyse (engl. Principal Component Analysis, PCA) [70] auf dem gleichen Trainingsdatensatz gefundenen Merkmalsvektoren. Einige der ersten (und damit wichtigsten) Hauptkomponenten (engl. Principal Components, PC) sind in Abbildung 5.7 zusammen mit der Rekonstruktion eines Beispielbildes zu sehen, die entstehen, wenn nur die n-ersten Hauptkomponenten als Dimensionen des Merkmalsraums verwendet werden [70, Kapitel 6].

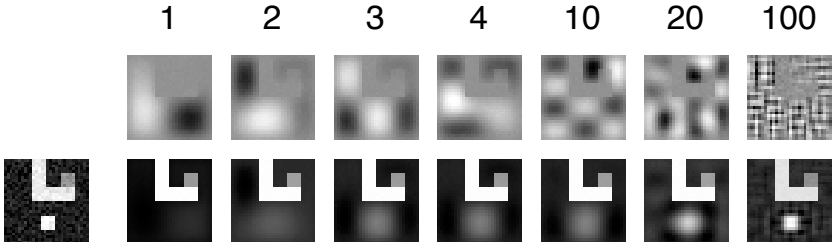


Abbildung 5.7: Visualisierung einiger der von der PCA gefundenen Hauptkomponenten (oben), wobei grau für betragsmäßig kleine Gewichte steht. Rekonstruktion des Bildes ganz links bei Verwendung der n-ersten Hauptkomponenten (unten).

Ergebnisse Standard-PCA bleibt weit vom Ziel entfernt, eine kompakte Repräsentation zu extrahieren; die ersten beiden der insgesamt 900 Hauptkomponenten erklären nur ca. 6% der Varianz in den Trainingsdaten. Das ist erstaunlich wenig; in einem Objekt- oder Gesichtserkennungsproblem würden die ersten beiden Hauptkomponenten eher 30% der Varianz erklären [82]. Die Verwendung nur dieser zwei wichtigsten Hauptkomponenten führt dementsprechend zu einer mit starkem Verlust behafteten

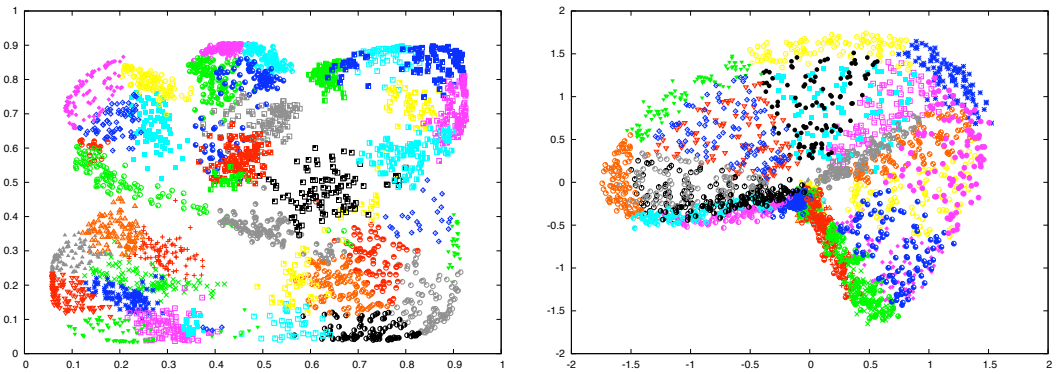


Abbildung 5.8: Vergleich der zweidimensionalen Merkmalsräume, die mit tiefem Lernen (links) und der Hauptkomponentenanalyse (rechts) gefunden wurden.

Kodierung, was sich in dem hohen Rekonstruktionsfehler von 15.8 gegenüber dem Rekonstruktionsfehler von 7.3 bei Verwendung der zwei vom Autoencoder gefundenen Dimensionen niederschlägt und nur knapp über dem Fehler des Durchschnittsbildes liegt.

Die Projektion der Daten in den von den beiden ersten Hauptkomponenten aufgespannten Raum ist in Abbildung 5.8 zu sehen. Im Vergleich zur Einbettung des tiefen Encoders gibt es deutlich mehr Überschneidungen zwischen den Clustern der einzelnen Klassen. Probleme hat die PCA auch insbesondere mit den Objekten innerhalb des kleinen Zielraums. Wie in Abbildung 5.7 zu sehen ist, gehen die Pixel innerhalb des Raums kaum in die ersten beiden Hauptkomponenten ein. Eine nur auf diesen beiden Komponenten basierende Einbettung in den Merkmalsraum ist daher an dieser Stelle des Labyrinths praktisch "blind". Diese Blindheit erklärt die freie Fläche im dritten Quadranten der Darstellung des Merkmalsraums in Abbildung 5.8. Ein Großteil der Daten landet nahe des Ursprungs und wird nicht ausreichend auseinander gezogen. Das Trainieren eines Klassifikators (Netzstruktur wie im vorigen Experiment) auf diese suboptimale Einbettung der Daten in den durch die zwei ersten Hauptkomponenten aufgespannten Merkmalsraum führt zu einer Klassifikationsrate von nur 39.58% auf den Testdaten. Es müssen einige weitere Hauptkomponenten zur Konstruktion eines höherdimensionalen Merkmalsraums verwendet werden, bis die Klassifikationsrate auf der zweidimensionalen, mittels DL gefundenen Kodierung wirklich übertroffen wird (siehe Tabelle 5.4).

Diskussion Einen Anhaltspunkt für das schlechte Ergebnis der PCA in dieser Problemstellung liefert das in Abbildung 5.9 dargestellte, vereinfachte Beispiel. Offensichtlich haben alle drei in der Abbildung dargestellten Bilder bezüglich der sie repräsentierenden Vektoren den gleichen (euklidischen) Abstand voneinander. Die Ähnlichkeit bezüglich der Position des weißen Flecks im Bild kann in dieser (üblichen) Form der Kodierung der Bilder nicht über ähnliche Werte innerhalb der einzelnen Dimensionen

Tabelle 5.4: Vergleich der Ergebnisse PCA / tiefer Autoencoder (Spalte DL 2D). Aufgeführt sind der durchschnittliche Rekonstruktionsfehler pro Testbild (RE) und der Anteil der korrekt klassifizierten Testbilder (CR) in der Klassifikationsaufgabe.

Kodierung →	DL 2D	2 PCs	4 PCs	10 PCs	20 PCs
RE	7.3	15.80	14.91	12.59	9.72
CR	80.10%	39.58%	70.80%	88.29%	98.61%

geschehen. Auch die Analyse der Kovarianzen zwischen den Dimensionen führt zu keinen Erkenntnissen, da es im dargestellten Fall keine "überlappenden" Objektflächen gibt. Die gewünschte Ähnlichkeit beruht vielmehr auf der nur implizit vorhandenen Nähe zwischen den orthogonalen Dimensionen der Bildvektoren. Auf Linearkombinationen der Dimensionen beruhende Methoden wie PCA haben daher Probleme damit, einen kompakten Merkmalsraum mit sehr viel weniger Basisvektoren (bzw. Dimensionen) als möglichen Positionen zu konstruieren – insbesondere, wenn die Objekte klein sind und es, wie im Labyrinthproblem zwischen Objekten inner- und außerhalb des Zielraums zu wenigen bzw. keinen Überlappungen kommt.

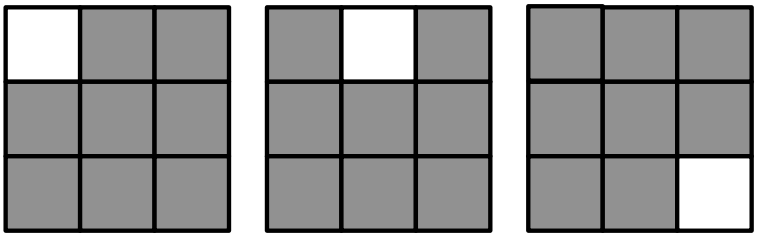


Abbildung 5.9: Die drei dargetstellten Bilder mit 3×3 Pixeln können, einer der üblichen Kodierungen für Grauwerte folgend, durch die drei 9-dimensionalen Vektoren $(1, \frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$, $(\frac{1}{2}, 1, \frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ und $(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}, 1)$ repräsentiert werden. Unter jeder beliebigen L^P-Norm sind diese Vektoren gleichweit voneinander entfernt, obwohl hinsichtlich der Position des weißen Rechtecks Bild 1 offensichtlich näher an Bild 2 als an Bild 3 ist.

Die tiefen Autoencoder sind dagegen in der Lage, Aussehen und Position des Objekts getrennt voneinander zu repräsentieren und den nichtlinearen Zusammenhang zwischen den Dimensionen auch automatisch besser zu erfassen. Das konkrete Aussehen des Objekts wird dabei vornehmlich in den Gewichten der äußeren Schichten des Encoder-Decoder-Netzes kodiert, während die tieferen Schichten eher mit der abstrakteren Position des Objekts arbeiten und diese schließlich in eine zweidimensionale Kodierung überführen. Das Resultat ist im hier untersuchten Grid-World-Beispiel eine nichtlineare, bedeutungsvolle und sehr kompakte Einbettung.

5.3.4 Rückwärtspropagieren des Gradienten in den Encoder

Eine Möglichkeit, die erlernten Einbettungen weiter zu verbessern, ist es, nicht nur den Autoencoder auf die Rekonstruktion der Daten zu trainieren und dann einen Encoder abzuleiten, sondern auch zusätzliche Informationen aus der anschließend zu erlernenden Aufgabe in die Anpassung der Encodergewichte einfließen zu lassen. In der Kombination des im Autoencoder auf die Rekonstruktion trainierten Encoders mit einem Task-Netz, das auf eine spezielle Klassifikations- oder Regressionsaufgabe trainiert wird, kann der Gradient $\nabla E/\nabla z$ des Fehlers E des Task-Netzes bezüglich der Eingaben z in das Task-Netz auch für die Optimierung der Einbettung in den Merkmalsraum genutzt werden. Dies geschieht, indem der Gradient des Fehlers in das Encodernetz eingespeist und dann weiter zurück propagiert wird, um auch dort zu einer Anpassung der Gewichte verwendet zu werden [61]. Die Gewichte werden im Encoder dabei so angepasst, dass die Ausgabe z des Encoders in die Richtung $-\nabla E/\nabla z$ bewegt wird, in der der Fehler des Task-Netzes kleiner wird. Auf diese Weise entsteht eine direkte Kopplung zwischen dem Merkmal extrahierenden Encoder und dem die "eigentliche" Lernaufgabe lösenden Netz. Beide Netze können in dieser Verknüpfung als ein großes Netz betrachtet werden (siehe auch schematische Zeichnung in der ersten Zeile von Abbildung 5.10).

Training auf die Position Dieses Vorgehen wird zunächst in der bereits beschriebenen Klassifikationsaufgabe getestet. An den auf den Rekonstruktionen voll austrainierten Encoder aus Abschnitt 5.3.2 wird ein weiteres Netz gehängt. Der Schlüssel für gute Ergebnisse ist die Verwendung eines sehr kleinen Netzes mit nur drei versteckten Neuronen und damit bei weitem zu beschränkter Ausdrucksstärke, um die Abbildung vom ursprünglichen, stark gefalteten Merkmalsraum (siehe Abbildung 5.5) auf die Positionskoordinaten zu erlernen. In der überwachten Lernphase wird der Fehler nach den ersten 20 Epochen, die zum groben Vortraining der Gewichte des Task-Netzes dienen, auch in das Encodernetz zurückpropagiert und dort für Gewichtsänderungen verwendet.

Das kombinierte Netz erreicht eine eindrucksvolle Klassifikationsrate von 99.46% auf den Testbildern gegenüber den bei Verwendung eines festen Merkmalsraums erzielten 80.10%. Noch erstaunlicher ist aber die erzielte Qualitätsverbesserung der Einbettung während des Trainingsvorgangs. Die rechte Spalte der Abbildung 5.10 zeigt die graduelle Verbesserung der Topologie in der Kodierungsschicht (im kombinierten Netz die dritte Schicht von hinten). Die Topologie wird auseinandergefalltet und entzerrt, bis zu einem nahezu perfekten Ergebnis mit nur wenigen Deformationen und keinen Überschneidungen. In der letzten Zeile ist der resultierende Merkmalsraum dargestellt. Es ist deutlich zu sehen, dass nicht nur die Anordnung der Daten, sondern auch die Trennbarkeit der Cluster deutlich verbessert wird. Zwischen den einzelnen Clustern befinden sich nun deutlich sichtbare Abstände, die so nicht aus der Position des Agenten herrühren, sondern durch das Training auf die zugrunde liegenden Klassenlabel erzwungen werden. In der linken Spalte der Abbildung ist zudem zu erkennen, wie sich der absolute Fehler der Projektion der Merkmalsvektoren zurück auf die Zellkoordinaten entwickelt.

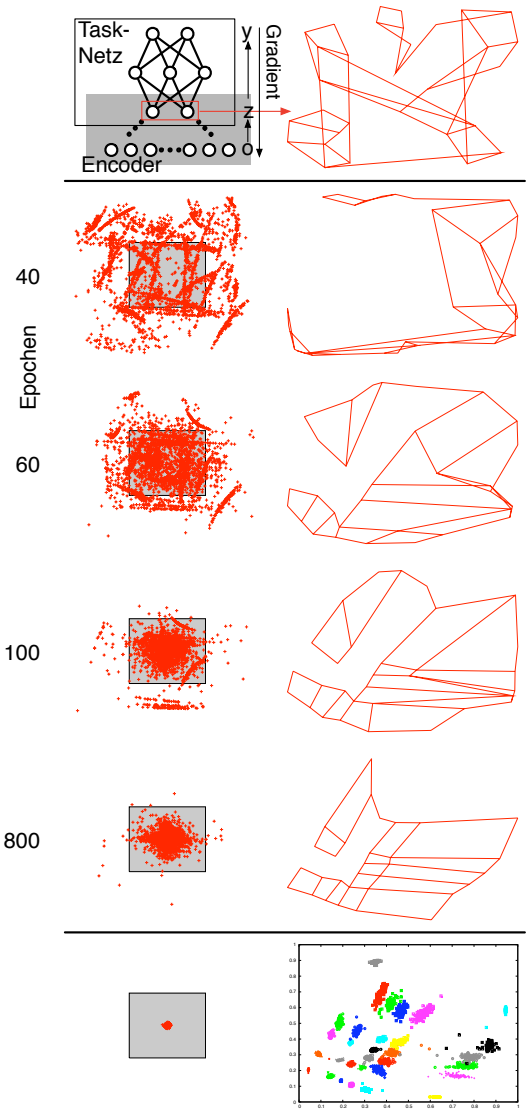


Abbildung 5.10: Anpassung der Encodergewichte beim Training auf eine überwachte Lernaufgabe. Oben: Skizze des Versuchsaufbaus (links). Die globale Topologie wird anfangs nicht gut im Merkmalsraums erfasst (rechts). Mitte: Topologie (rechte Spalte) im Merkmalsraum und absolute Differenz zwischen den Zielwerten und der Netzausgabe auf den Testbildern (linke Spalte). Das graue Rechteck markiert den Fehlerbereich, der nicht zu einer Fehlklassifikation führt. Unten: Der absolute Fehler auf den Trainingsbildern (links) und die Verteilung der Testbilder im Merkmalsraum nach 800 Episoden (rechts).

Merkmalsvektoren, die anfangs außerhalb der grauen Fläche[1] liegen und deshalb falsch klassifiziert werden, wandern mit der Verbesserung des Merkmalsraums immer näher

[1] entspricht der relativen Ausdehnung der Zelle gemessen von ihrem Mittelpunkt

an den Mittelpunkt der richtigen Zelle. In der letzten Zeile ist zum Vergleich die Lage der Trainingsbilder dargestellt.

Mit Hilfe dieser Technik ist es nicht nur möglich, eine bereits gute Kodierung weiter zu optimieren, sondern auch eine sehr schwache initiale Kodierung auf ein ordentliches Niveau zu verbessern. Ein zunächst auf nur 775 ungleichmäßig verteilte Bilder trainierter Encoder (durchschnittlicher Rekonstruktionsfehler: 12.0) kann hinsichtlich der Trennbarkeit der Cluster und der Topologie deutlich verbessert werden (siehe Abb. 5.11), so dass die Klassifikationsrate auf 80.87% (von 59.61% bei fixierten Encodergewichten) ansteigt.

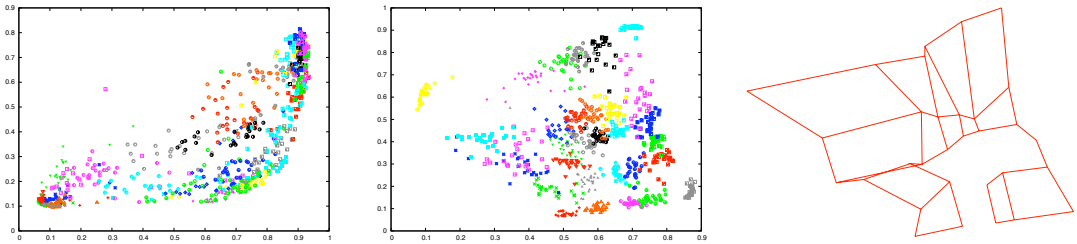


Abbildung 5.11: Verbesserung eines Encoders, der von einem schlecht trainierten Autoencoder abgeleitet wurde, im Training auf eine überwachte Lernaufgabe. Merkmalsraum nach dem Feintraining (links) und nach dem Training auf die Klassifikation (Mitte). Topologie nach dem Training auf die Klassifikation (rechts).

Training auf die optimale Zustandswertfunktion In einem abschließenden Experiment wird ein vortrainierter Encoder zusammen mit einem Task-Netz auf die optimale V-Funktion V^* der Grid-World trainiert. Diese kann für die vorliegende Grid-World leicht per Hand berechnet werden und müsste im späteren Einsatz in DFQ als $\hat{V}^i$ iterativ angenähert werden. Während bei fixen Encodergewichten unter Einsatz eines Task-Netzes mit optimaler Topologie lediglich eine Klassifikationsrate (Kosten auf einen halben Schritt Genauigkeit korrekt geschätzt) von 78.65% zu erzielen ist, führt die Anpassung der Encodergewichte mit einem Task-Netz mit 12 versteckten Neuronen zu einer Klassifikationsrate von 99.54%.

Im Versuch mit Adaptation der Encodergewichte wird der Merkmalsraum stark auf die Erfordernisse der überwachten Lernaufgabe angepasst und vollständig zu einem schmalen Band kontrahiert (siehe Abb. 5.12). An einem Ende des dargestellten Bandes liegen die Zustände, deren Wert 0 ist, am anderen Ende die Zustände mit Wert -10. Zum Beispiel werden die beiden markierten Zellen, die in der zugrunde liegenden Grid-World geometrisch nahe beieinander liegen, aber aufgrund der Wand zwischen ihnen stark differierende Zustandswerte haben (-1 und -9), zu den unterschiedlichen Enden des Bandes bewegt und dort mit Observationen anderer Zellen mit gleichem V-Wert zusammengelegt.

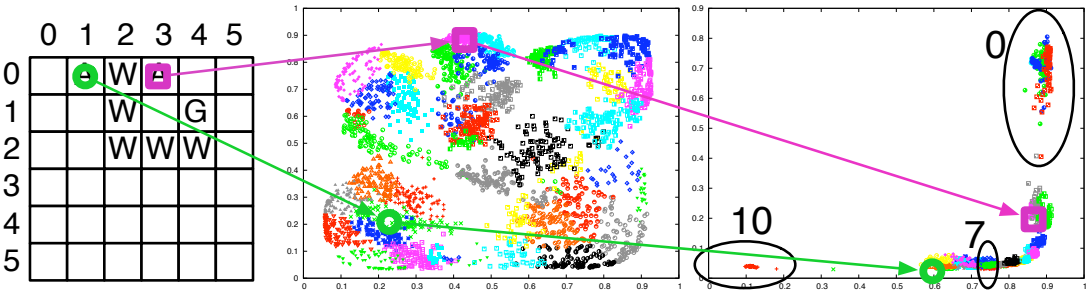


Abbildung 5.12: Position der Zellenmittelpunkte $(1.5, 0.5)$ und $(3.5, 0.5)$ im Labyrinth (links), Merkmalsraum nach dem Feintraining (Mitte) und nach dem Training auf die optimale Wertfunktion (siehe Text).

Diskussion Die hier diskutierten Techniken sind eine gute Möglichkeit, einen Merkmalsraum weiter zu verbessern, wenn neben den Observationen auf weitere Informationen zurückgegriffen werden kann. Können die Observationen in einer gegebenen Problemstellung zum Beispiel per Hand mit den internen Systemzuständen versehen werden, ist das beschriebene Vorgehen direkt anwendbar. Nach der Verbesserung der Merkmalsextraktion ist die Kenntnis der Systemzustände in der späteren Anwendungsphase nicht mehr nötig. Können dagegen, zum Beispiel wegen des hohen Aufwands, nur für einen Teil der Beobachtungen die richtigen Systemzustände bestimmt werden, kommen der Einsatz von Multi-Task-Lernen [22] und Semi-Supervised Learning in Frage. Durch das Anfügen eines Task-Netzes an die Kodierungsschicht kann der Encoder gleichzeitig auf die Rekonstruktion aller Daten im Autoencoder und auf die nur teilweise verfügbaren Klassenlabel trainiert werden.

Darüber hinaus ist es denkbar, eine initiale Kodierung direkt auf die Abspeicherung der Zustands-Wertfunktion anzupassen. Bei diesem Vorgehen würden lediglich Informationen verwendet, die durch das dynamische Programmieren selbst gewonnen werden. Die damit verbundene Hoffnung wäre, dass sich der Merkmalsraum ausgehend von den absorbierenden Zielzuständen an die RL-Aufgabe anpasst, an denen der tatsächliche, optimale Zustandswert nach der ersten DP-Aktualisierung bekannt ist.

5.3.5 Verkleinerung der Trainingsmenge führt zu Problemen

Die bisherigen Ergebnisse belegen, dass DL prinzipiell in der Lage ist, einen sehr niedrigdimensionalen Merkmalsraum aus den Bilddaten eines Navigationsproblems zu konstruieren, der für die Anwendung von RL geeignet ist. Allerdings besteht ein wesentliches Problem darin, dass für eine gute Einbettung und stabile Ergebnisse eine relativ große Menge Daten benötigt wird. Zudem ist auch die gleichmäßige Verteilung der Daten von Bedeutung.

Ergebnisse In Tabelle 5.5 ist zu sehen, wie sich der Rekonstruktionsfehler (Zeile RE) und die Klassifikationsraten (CR) in der Koordinaten-Klassifikationsaufgabe ent-

wickeln, wenn die Menge der Daten abgesenkt und die Verteilung verschlechtert wird. Die 3 100 mit gleichmäßiger Verteilung gezogenen Trainingsbilder entsprechen einer 4-fachen Abdeckung aller 775 freien Positionen, die der Agent im Bild einnehmen kann. Die Ergebnisse verschlechtern sich schnell, wenn die Anzahl der im Training zur Verfügung stehenden Bilder abgesenkt und die perfekte Verteilung aufgegeben wird[1]. Im Falle des Besuchs von immer noch 60% aller möglichen Positionen kommt es bereits zu einer schwachen Klassifikationsrate von ca. 27%. Die große Mehrheit der von diesem schlecht trainierten Autoencoder rekonstruierten Testbilder zeigt einen oder mehrere Agenten an völlig falschen Positionen.

Tabelle 5.5: Rekonstruktionsfehler (RE) und Klassifikationsraten beim Training mit (CR Adaptiv) und ohne (CR Fixiert) Anpassung der Encodergewichte beim Einsatz unterschiedlich großer Trainingsmengen. Nur Experimente mit einer Abdeckung > 1 verwendeten eine gleichmäßige Verteilung der Trainingsbilder über die möglichen Positionen.

Abdeckung →	0.2	0.6	1	2	4
RE	27.58	18.9	12.0	9.4	7.3
CR Fixiert	24.52%	27.31%	59.61%	61.42%	80.10%
CR Adaptiv	25.81%	40.86%	80.87%	91.59%	99.46%

Diskussion Diese hohen Anforderungen an die Menge der Samples und deren Verteilung stellen in der Praxis ein Problem in der Kombination mit RL dar. Zwar lag die Abdeckung des Observationsraums in den Experimenten mit maximal 4/1 schon deutlich niedriger als beim MNIST-Datensatz und in den Experimenten von Gordon und Ernst, die Ergebnisse reichen aber nicht für eine dateneffiziente Anwendung in DFQ. Von einem lernenden Agenten würde man erwarten, dass er beim Erlernen einer Strategie über die einzelnen Positionen zu einem gewissen Grad generalisiert und nicht jede einzelne Stelle mehrmals besucht haben muss, um eine gute Abschätzung der Kosten vornehmen zu können. Die Generalisierung bei der Strategie ist im visuellen Reinforcement Lernen aber nur möglich, wenn sie auch vom Wahrnehmungsteil getragen wird, also der Autoencoder bereits gute Merkmalsvektoren liefert, wenn noch gar nicht alle Positionen besucht wurden und die Abdeckung deutlich kleiner als 1 ist. Außerdem ist es in vielen sequentiellen Entscheidungsproblemen unmöglich, eine gleichmäßige Verteilung der Daten über die Zustände zu erreichen, insbesondere natürlich, wenn noch kein adäquater Autoencoder vorhanden ist und die verschiedenen Zustände noch gar nicht unterschieden werden können. Wie bereits diskutiert, häufen sich zudem die beobachteten Zustände typischerweise anfangs um die Start- und Attraktorzustände (z.B. Tal im Mountain-Car), während im weiteren Lernverlauf eine Häufung um die Zielzustände bzw. entlang der optimalen Trajektorien angestrebt wird. Aus diesem Grund wird im

[1]Für alle Abdeckungen ≤ 1 werden in allen Experimenten die Agentenpositionen, von denen Observationen erzeugt werden, in einem Urnenexperiment mit zurücklegen gezogen.

Folgenden der Einsatz der Faltungskernel erprobt, durch deren Anwendung die Qualität der gefundenen Einbettung auch unter schwierigen Bedingungen weiter verbessert werden kann.

5.3.6 Faltungskerne erleichtern das Training

Im vorhergehenden Experiment wurde festgestellt, dass die Rekonstruktionsfehler und Klassifikationsfehler schnell ansteigen, wenn die Trainingsmengen in für RL interessanten Bereich von Abdeckungen kleiner 1 abgesenkt werden. Hier bietet der Einsatz von Faltungskernen eine Verbesserungsmöglichkeit.

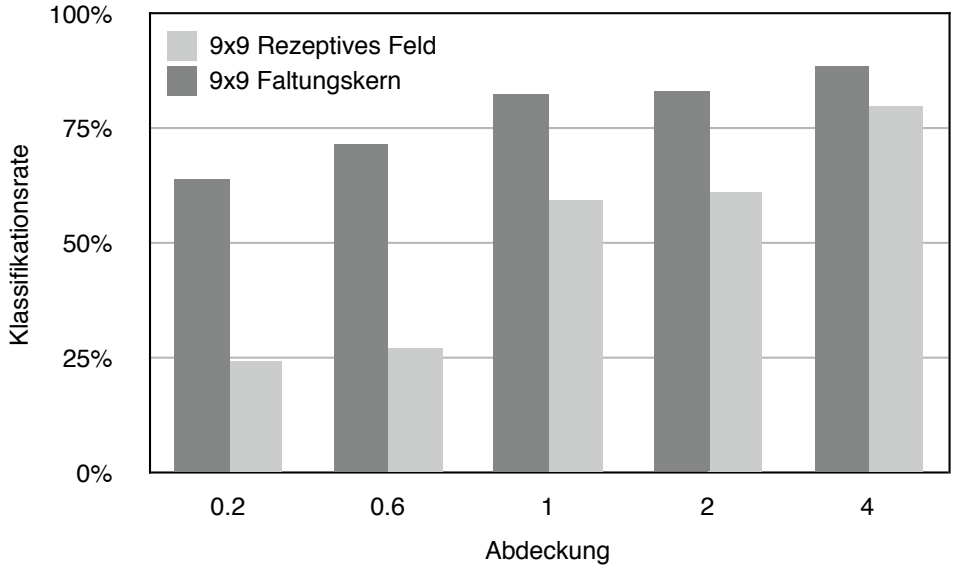


Abbildung 5.13: Vergleich der Klassifikationsraten (fixierte Encodergewichte) bei Verwendung eines Faltungskerns und rezeptiver Felder mit je 9×9 Neuronen.

Vergleich zwischen rezeptiven Feldern und Faltungskernen In Abbildung 5.13 sind die Ergebnisse eines Vergleichsexperiments unter Anwendung geteilter Gewichte dargestellt. Es wird das gleiche Netz wie in den vorhergehenden Experimenten verwendet, nur dass die rezeptiven Felder gemeinsame Gewichte nutzen. Gegenüber den Ergebnissen bei Verwendung unabhängiger rezeptiver Felder führt die Verwendung von Faltungskernen zu durchweg besseren Ergebnissen. Gerade bei den Versuchen mit sehr wenigen Daten kommt es zu den deutlichen Verbesserungen. Selbst mit einer Abdeckung von nur 20%, das sind in der Summe 155 Trainingsbilder, sind die Ergebnisse immer noch besser als die Ergebnisse des Netzes mit rezeptiven Feldern bei 200% Abdeckung.

Untersuchung unterschiedlicher Kerngrößen Wie eine Versuchsreihe mit unterschiedlichen Kerngrößen zeigt (siehe Tabelle 5.6), sind die Klassifikationsraten, unabhängig von der Wahl der Größe des Kerns, durchweg besser als die Ergebnisse, die mit den rezeptiven Feldern auf dem gleichen Datensatz erzielt werden. Die Klassifikationsraten und Rekonstruktionsfehler bewegen sich für alle Kerne in ähnlichen Größenordnungen. Die absolut besten Ergebnisse, gerade bei der Rekonstruktion und der Klassifikation mit fixierten Encodergewichten, erzielen die Netze mit dem kleinsten Kern von 3×3 Neuronen, bei sehr wenigen Daten scheint der Kern mit 9×9 Neuronen einen leichten Vorteil im Falle der fixierten Encodergewichte zu haben.

Tabelle 5.6: Resultate mit Faltungskernen unterschiedlicher Größe. Rekonstruktionsfehler (RE) und Klassifikationsraten beim Training mit (CR Adaptiv) und ohne (CR Fixiert) Anpassung der Encodergewichte. Nur Experimente mit einer Abdeckung > 1 verwendeten eine gleichmäßige Verteilung der Trainingsbilder über die möglichen Positionen.

Versuch	Kern	0.2	0.6	1	2	4
RE	3x3	14.57	8.13	6.24	5.61	4.62
	5x5	13.39	9.28	8.26	6.80	6.06
	7x7	13.18	9.27	7.96	6.43	4.93
	9x9	12.38	9.42	7.77	7.55	6.98
CR Fixiert	3x3	54.19%	76.56%	80.13%	91.16%	92.81%
	5x5	48.39%	69.46%	72.77%	79.10%	82.97%
	7x7	47.10%	71.18%	70.97%	83.68%	91.29%
	9x9	63.87%	71.83%	82.71%	82.97%	88.77%
CR Adaptiv	3x3	72.90%	89.03%	97.29%	100%	100%
	5x5	61.94%	84.30%	92.65%	99.35%	100%
	7x7	72.90%	85.81%	94.71%	99.42%	100%
	9x9	64.52%	87.75%	92.39%	99.87%	99.90%

Untersuchung der Robustheit Die Faltungskerne erhöhen auch die Robustheit der Klassifikationsergebnisse gegenüber dem Pixelrauschen. Das Training von Autoencodern mit 9×9 Faltungskernen auf vier Datensätzen mit je 3 100 Trainingsbildern und unterschiedlichen Parametern für die Stärke des Bildrauschens führt zu den in Abbildung 5.14 dargestellten Ergebnissen. Die Rate bleibt für $\sigma \leq 0.3$ relativ stabil und oberhalb der mit den rezeptiven Feldern auf dem Datensatz mit $\sigma = 0.1$ erzielten Ergebnis (vgl. Tabelle 5.5). Probleme bereiten bei erhöhtem Rauschen insbesondere die Bilder, in denen der Agent nur halb im Bild ist (siehe z.B. Bild 3 in Abb. 5.14). Erst mit einem Wert $\sigma > 0.3$, der weitab dessen liegt, was in der Praxis bei einer realen Kamera anzutreffen ist, fällt die Klassifikationsrate deutlich ab.

Diskussion Autoencoder mit Faltungskernen benötigen wesentlich weniger Trainingsdaten, um die gleichen Ergebnisse wie identisch aufgebaute Netze mit rezeptiven Feldern zu erzielen. Auch bei einer ungleichmäßigen Verteilung der Trainingsdaten (untersucht

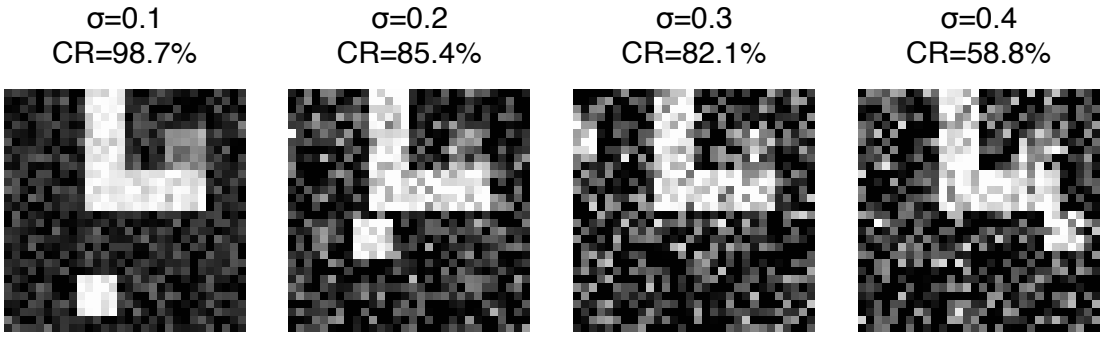


Abbildung 5.14: Beispielbild und erzielte Klassifikationsrate (CR) bei Erhöhung der Standardabweichung des Rauschens auf $\sigma = 0.1, 0.2, 0.3, 0.4$.

in Abdeckungen ≤ 1) werden noch gute Ergebnisse erzielt. Durch diese Technik wird später das Erlernen brauchbarer Einbettungen auch in der Anfangsphase der Exploration ermöglicht. Die Größe der eingesetzten Kernel spielt dabei eine eher untergeordnete Rolle. Solange sich das Rauschen in Bereichen bewegt, die in der Praxis realistisch sind ($\sigma < 0.2$), erzielen die Autoencoder mit Faltungskernen auch unter diesen erschwerten Bedingungen relativ stabile Ergebnisse mit Klassifikationsraten deutlich über 80%.

5.3.7 Anwendung auf reale Bilddaten

Die in ausführlichen Versuchen auf den synthetischen Bilddaten entwickelte und getestete Architektur mit geteilten Gewichten soll nun abschließend auf realen Bilddaten getestet werden. Dazu wird ein Datensatz von 2 000 realen Trainingsbildern verwendet, die vom im Kapitel 7 eingehend behandelten Carrerabahn-System [74] stammen. Sie zeigen einen Spielzeugwagen, der mehrere Runden auf der Bahn fährt. Die Trainingsbilder wurden zusammen mit 2 000 weiteren Testbildern mit 60 Hz innerhalb von 66 Sekunden Fahrtzeit aufgenommen.

Netzarchitektur und Trainingsprozedur Um die Bilder mit 64×68 Pixeln zu verarbeiten, wird ein zehnschichtiges Encodernetz mit der bereits bekannten, weitestgehend generischen Architektur mit Reduktionsfaktor 2 bis hin zur Kodierungsschicht mit zwei Neuronen verwendet. In den beiden äußersten Verbindungsschichten werden 9×9 Faltungskerne verwendet, die nachfolgenden Schichten sind jeweils voll verbunden. Dieser Encoder besitzt in der Summe über 800 000 Verbindungen. Nach dem Abschluss des schichtenweisen Vortrainings mit je 50 Trainingsepochen je Schicht, werden mehrere Tausend Epochen Finetuning durchgeführt, wobei 10 Mini-Batches zum Einsatz kommen. Die Verwendung der Mini-Batches ist keinesfalls notwendig, führt auf diesem Datensatz aber zu einem minimal schnelleren Sinken des Rekonstruktionsfehlers.

Rekonstruktion Eine Begutachtung der Rekonstruktionen der Testbilder zeigt, dass der Wagen durchweg an der richtigen Stelle eingezeichnet wird, was es erfordert, dass

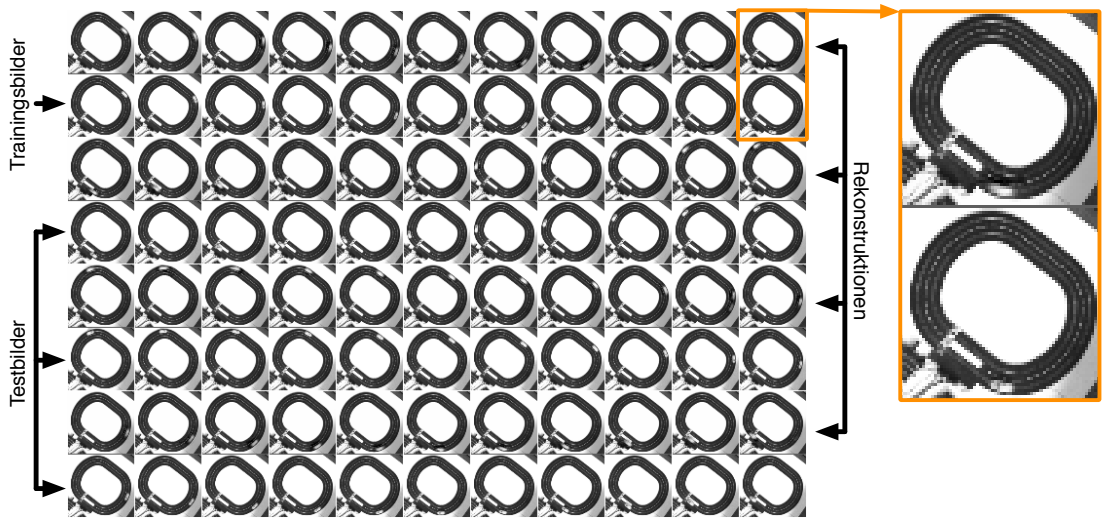


Abbildung 5.15: Rekonstruktionen des auf 2000 Bildern trainierten Autoencoders.

seine Position im Originalbild richtig erfasst und in den zwei innersten Neuronen richtig kodiert wird. In Abbildung 5.15 sind einige der erzeugten Rekonstruktionen nach 12 000 Trainingsepochen zu sehen. Im vergrößerten Ausschnitt ist erkennbar, dass der Wagen sehr genau an der richtigen Stelle auf der Strecke eingezeichnet wird, dabei aber einige feine Konturen verloren gehen. Dies ist auf die Verwendung nur eines Kerns zurückzuführen, der die feinen Details des Wagens nicht in allen Orientierungen exakt erfassen kann.

Visuelle Analyse des Merkmalsraums In Abbildung 5.16 (links) sind die Merkmalsvektoren der 2 000 Testbilder und der Verlauf einer einzelnen Runde im Merkmalsraum dargestellt. Alle möglichen Wagenpositionen auf der Strecke liegen entlang eines schmalen Bandes im Merkmalsraum.

Die eine Hälfte der Strecke wurde dabei im linken Teilbereich des Merkmalsraums extrem gut auseinander gezogen. Am Anfang und Ende dieser Teilstrecke gibt es jeweils einen großen Sprung an eine entfernte Stelle im Merkmalsraum, wo sich ein weiteres Teilstück anschließt. In diesem zweiten Teilstück gibt es zwei Überschneidungen des Bandes mit sich selber. Die beiden Sprünge und die Überschneidungen stellen ein Problem in der Repräsentation dar, da es hier zu Verwechselungen von Positionen kommen kann. So sind entlang des diagonal von links oben nach rechts unten verlaufenden Sprungs einzelne Merkmalsvektoren zu sehen, die zu Bildern "zwischen" den Positionen vom Anfang und Ende des Sprungs gehören, die aber sehr nahe an dem übersprungenen Teilbereich des Bandes liegen und so in einer Klassifikation oder beim Erlernen einer Wertfunktion einer falschen Streckenposition zugeordnet werden könnten.

Abgesehen von diesen vier lokal beschränkten Problemstellen ist die lineare Anordnung der Observationen entlang des Streckenverlaufs aber klar im Merkmalsraum zu

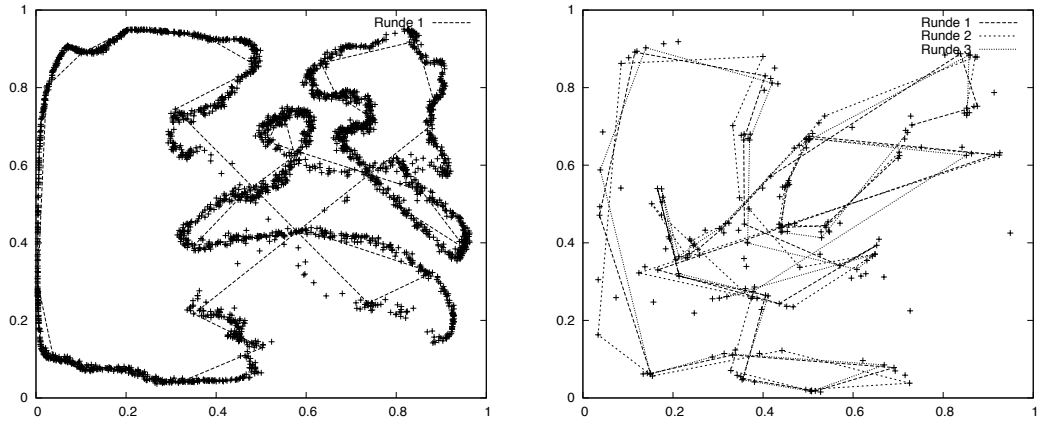


Abbildung 5.16: Einbettung der Testbilder im auf Basis von 2000 Bildern (links) und 200 Bildern (rechts) gewonnenen Merkmalsraum. Im Bild links wurden die Bilder einer einzelnen vollständigen Runde mit einer Line verbunden, im Bild rechts wurden drei Runden eingetragen.

erkennen. Die geringe Breite des Bandes und seine scharfen Grenzen deuten auf eine sehr gute Robustheit gegenüber dem Bildrauschen und anderen Störeinflüssen hin. Nur sehr wenige Testbilder – quasi nur ein paar Bilder entlang der beiden problematischen Sprünge – liegen überhaupt außerhalb des Bandes.

Entwicklung des Merkmalsraums Die in Abbildung 5.17 dargestellte Auswertung wirft ein wenig Licht auf die Frage, wie es zu solchen lokal unterschiedlich guten Organisationen des Merkmalsraums und zu den beobachteten Sprüngen kommen kann. Es lohnt sich eine ausführliche Besprechung der aufgezeichneten Entwicklung, da sie einen Einblick in die Vorgänge während des Trainings des tiefen Netzes liefert.

Nach Abschluss des Vortrainings (Epoche 0) wird fast immer nur das Durchschnittsbild rekonstruiert, was sich im hohen Fehler – in der oberen Reihe der Abbildung weiß dargestellt – entlang des gesamten Streckenverlaufs niederschlägt. Gleichzeitig ist aber in der Darstellung der Merkmalsvektoren zu erkennen, dass es bereits eine Struktur im Merkmalsraum gibt. Insbesondere wurde bereits ein Teilabschnitt am linken Rand des Merkmalsraums sehr gut auseinander gezogen, während ein anderer Teilabschnitt in der rechten Hälfte des Merkmalsraums stark in sich selbst verschlungen ist.

Gleich zu Beginn des Feintrainings in den ersten 60 Epochen werden die Gewichte so geändert, dass der gut erfasste Teilbereich weiter auseinander gezogen und ihm ein noch größerer Raum eingeräumt wird, während der nicht so gut erfasste Teilbereich sogar noch "zusammenfällt" und auf einen kleinen Fleck rechts oben in der Ecke abgebildet wird. Zu begründen ist dies damit, dass die Merkmalsvektoren im initial gut erfassten Bereich leicht voneinander zu trennen sind und dass für sie der Rekonstruktionsfehler schnell gesenkt werden kann, insbesondere wenn man sie noch besser separiert. Da für die Merkmalsvektoren im Knäuel keine großen Verbesserungen bei den Rekonstruktionen

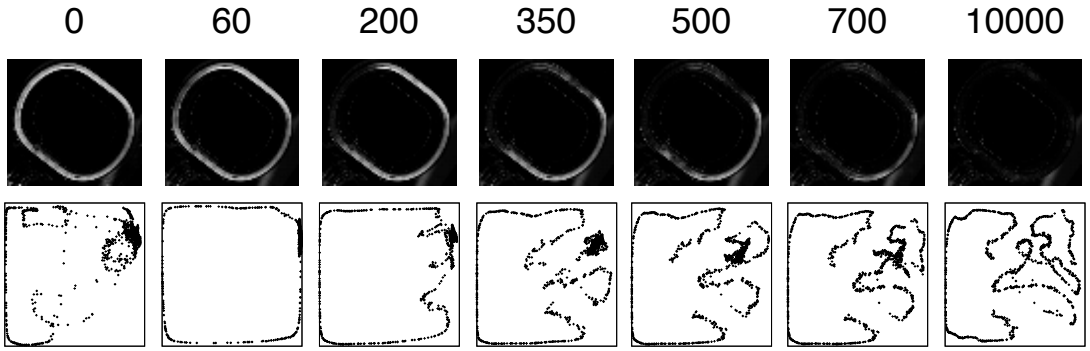


Abbildung 5.17: Entwicklung des Merkmalsraums (unten) und des Rekontruktionsfehlers im Bildraum (oben) nach $0, 60, 200, 350, 500, 700, 10000$ Epochen Feintraining.

ohne größere Umordnungen dieser Merkmalsvektoren erzielt werden können, scheinen sie anfangs nur einen kleinen Einfluss auf die Gewichtsänderungen auszuüben. Bis zur Epoche 200 wird deutlich, dass der Rekonstruktionsfehler zuerst in der Kurve der Strecke links oben erkennbar gesenkt werden kann und sich dann quasi von dort aus in beide Richtungen der Strecke ausbreitet. Dies ist tatsächlich genau der Bereich, der auf den von Beginn an gut im Merkmalsraum erfassten Streckenbereich fällt.

Im weiteren Trainingsverlauf wächst der Einfluss auf die Gewichtsänderungen, den die Merkmalsvektoren im Knäuel ausüben, in Relation zu den gut erfassten Merkmalsvektoren an, da der zu ihnen gehörende Rekonstruktionsfehler bereits erfolgreich minimiert wurde und nicht mehr wesentlich gesenkt werden kann. Jetzt werden die Merkmalsvektoren aus der Ballung herausgezogen, wie wenn man an den Fadenenden eines Wollknäuels zieht. Bei diesem Vorgang muss der Entfaltung dieser Teilstrecke immer mehr Platz im Merkmalsraum eingeräumt und der bereits gut repräsentierte Teilbereich notgedrungen etwas zusammengestaucht werden. Während dieser Entfaltungsprozedur reißen die ehemals verbundenen Teilstücke voneinander ab und es kommt zu den beiden besagten Sprüngen.

Am Ende der Prozedur wird der Wagen in allen Streckenbreichen richtig rekonstruiert. Das kleine hellere Dreieck, dass nach 10 000 Epochen noch unten rechts im letzten Fehlerbild von Abbildung 5.17 zu sehen ist, rührt von einem Schlaglicht her, dass aufgrund einer Änderung der Sonneneinstrahlung nur in einem Teil der Bilder auf die Strecke fällt.

Kleine Trainingsmenge Abschließend wird auch auf den realen Bildern ein Versuch mit einer kleinen Trainingsmenge durchgeführt. Verwendet werden 200 Bilder, die in knapp 5 Runden in unter 10 Sekunden Interaktion bei ca. 20 Hz aufgezeichnet werden können. Nach 4000 Epochen wird bereits ein nicht perfektes, aber akzeptables Rekonstruktionsergebnis erzielt. Nach manueller Auszählung werden über 80% der Testbilder richtig rekonstruiert, was umso eindrucksvoller erscheint, wenn man bedenkt, dass hier die über 1 Mio Gewichte des Autoencoders auf nur 200 Trainingsbeispielen eingestellt

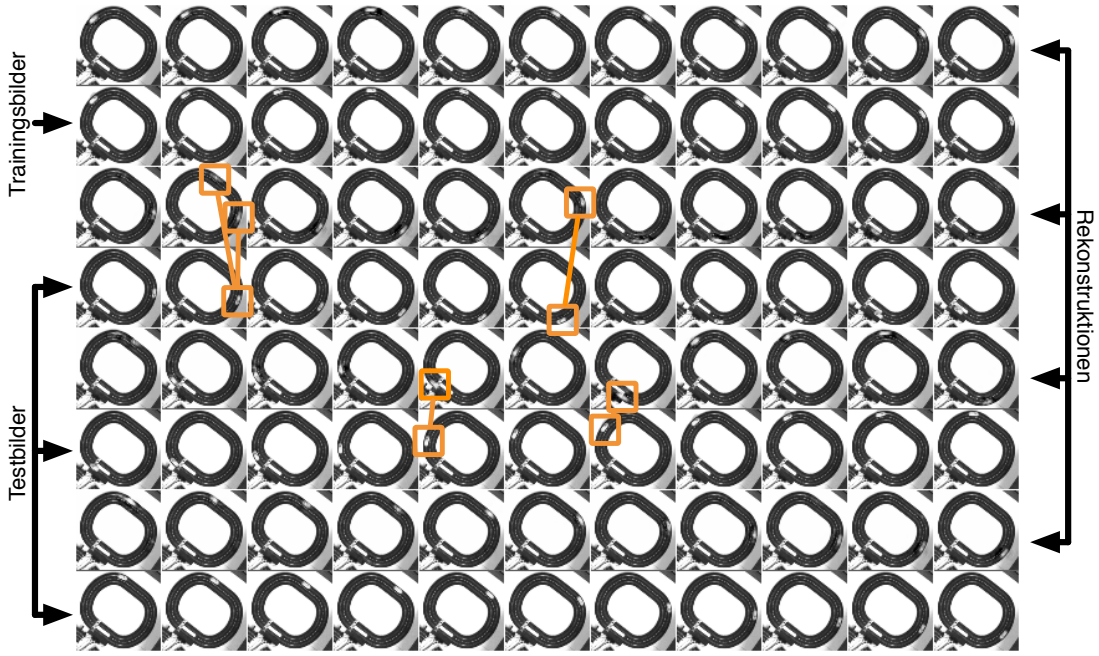


Abbildung 5.18: Rekonstruktionen des auf 200 Bildern trainierten Autoencoders.

werden mussten. In den fehlerhaften Rekonstruktionen wird der Wagen an der falschen Stelle eingezeichnet. Dies deutet auf ein "Verwechseln" der tatsächlich zu diesen Merkmalsvektoren gehörenden Positionen hin. Der Graph des Merkmalsraums in Abbildung 5.16) (rechts) zeigt ebenfalls die beginnende Formierung eines Bandes, aber es kommt zu mehr Überschneidungen als im anderen Experiment und es liegen auch mehrere Merkmalsvektoren außerhalb des Streckenverlaufs.

Diskussion Auch auf den realen Bildern bestätigen sich die Ergebnisse aus der Evaluation mit synthetischen Bildern. Ein auf 2 000 Trainingsbildern – diese können von der Carrerabahn in ca $35s$ gewonnen werden – trainierter Autoencoder ist in der Lage, den nur wenige Pixel großen Wagen zuverlässig im Bild zu detektieren, seine Position zu extrahieren und – abgesehen von vier problematischen, aber lokal begrenzten Überschneidungen und Sprüngen – sehr gut in dem aufgespannten, zweidimensionalen Merkmalsraum zu repräsentieren. Auch auf einer kleineren Trainingsmenge kann bereits eine ordentliche Einbettung erzeugt werden, die zumindest das Erlernen einer ersten, vorläufigen Strategie unterstützt.

In der Analyse der Entwicklung des Merkmalsraums im Training zeigt sich ein potenzielles Problem in praktischen Anwendungen. Eine anfänglich schlechte Anordnung der Merkmalsvektoren kann unter Umständen in der durch den Gradientenabstieg vollzogenen Optimierung nicht mehr grundsätzlich korrigiert werden und letztendlich zu bleibenden Faltungen und Überschneidungen führen. Es entsteht der Eindruck, dass ein

"Knäuel" im Finetuning noch besser entfaltet werden könnte, wenn mehr "Platz" – zum Beispiel in einer weiteren Dimension des Merkmalsraums – zur Verfügung stehen würde. Dieser Überlegung wird in Abschnitt 7.3 nachgegangen, wo für die RL-Experimente auch ein dreidimensionaler Encoder auf der Carrerabahn trainiert wird.

5.4 Diskussion und Zusammenfassung

Es wurde ein vereinfachter, nur auf MLPs beruhender Trainingsalgorithmus für tiefe Autoencoder vorgestellt und unter Ausnutzung moderner, paralleler Rechnerarchitekturen effizient implementiert. Dabei konnte die Laufzeit gegenüber der Standardimplementierung N++ um den Faktor 27 beschleunigt werden. Aufgrund der Verwendung gewöhnlicher MLPs ohne Besonderheiten wie symmetrische Verbindungsstrukturen oder flache RBMs war es möglich, sowohl rezeptive Felder als auch Faltungskerne (Shared-Weights) in den Trainingsablauf der tiefen Autoencoder zu integrieren. Die rezeptiven Felder, wie auch die Faltungskerne sind eine Möglichkeit, die Anzahl der zu trainierenden Gewichte zu verringern und gleichzeitig die Nachbarschaftsbeziehung zwischen den Pixeln auch explizit in der Netzarchitektur zu kodieren.

Im Rahmen der durchgeführten Evaluation wurde die prinzipielle Tauglichkeit der tiefen Netze für die Objektpositionserkennung in typischen RL-Benchmarkproblemen mit synthetischen Bildern und auf einem Satz realer Bilder untersucht und zumindest für diese Problemstellungen empirisch nachgewiesen. Die tiefen Netze haben sich dabei in einem direkten Vergleich mit einer in der Bildverarbeitung sehr gebräuchlichen, linearen Methode klar überlegen gezeigt. Mit der linearen Methode gelang es bei Weiterm nicht, die Bewegungen des Objekts vergleichbar zuverlässig in allen Bereichen des Labyrinths zu erfassen und so kompakt wie mit den Autoencodern zu repräsentieren.

In der Bewertung standen die eingangs in Abschnitt 5.1 aufgeworfenen Fragestellungen nach den Differenzierungs- und Generalisierungsmöglichkeiten, die sich dem Reinforcement Lerner im Merkmalsraum eröffnen und dem Umgang mit kleinen, ungleichmäßig verteilten Trainingsdatensätzen im Vordergrund. Zu diesen Punkten im Einzelnen:

1. **Zustandsidentifizierung und Generalisierung** Die visuelle Analyse der erzeugten Merkmalsräume sowie das als Probe durchgeführte Training auf eine Klassifikation führen zu durchweg positiven Bewertungen im Sinne der Kriterien 1 bis 3. Im Merkmalsraum bilden die Observationen, die den Agenten an ähnlicher Position zeigen, klar erkennbare Häufungen (Kriterium 3), die zumeist saubere Grenzen zu den Clustern benachbarter Agentenpositionen aufweisen (Kriterium 1 und 3). Das in der Bildformation eingeführte Zufallsrauschen führt schon zu einer sichtbaren Streuung der Merkmalsvektoren identischer Agentenpositionen – die Merkmalsvektoren liegen nicht exakt übereinandet, verhindert aber offensichtlich nicht die Herausformung klar erkennbarer Häufungen und steht auch guten Klassifikationsergebnissen nicht im Wege (Kriterium 2). Auf Basis der Merkmalsvektoren ist es zudem möglich, Klassifikationsaufgaben mit sehr guter Generalisierungsleistung zu erlernen. Das ist nur möglich, wenn die Merkmalsvektoren

unterschiedlicher Positionen trotz des Rauschens klar auseinandergehalten werden können (Kriterium 1 und 2). Eine gute Generalisierungsleistung kann zudem nur gelingen, wenn die Merkmalsvektoren bisher nicht gesehener Bilder ebenfalls richtig abgebildet werden, also eine Verallgemeinerung auf bisher nicht Gesehenes möglich ist (Kriterium 3).

Die Leistung des Netzes in der erlernten Positionsklassifizierung stellt gewissermaßen eine Baseline für die Performance in den späteren Lernexperimenten dar, wo auch eine Wertfunktion und zugehörige Strategie für das Labyrinth erlernt werden soll. Gelingt es nicht, einen Merkmalsvektor richtig zu klassifizieren, also auf die tatsächliche Position des Agenten im Labyrinth abzubilden, kommt dies quasi einer Verwechselung der Positionen gleich; der Agent wird im Fall einer solchen Zustandsverwechselung auch nicht den korrekten Zustandswert erlernen können. Im Prinzip muss zum Erlernen einer Wertfunktion für die Observationen im Grid-World-Problem implizit immer eine Klassifikation mit gelöst werden, um die richtigen verrauschten Bilder zusammenzufassen und auf den gemeinsamen optimalen Zustandswert abzubilden.

Hinsichtlich des vierten Kriteriums, der geeigneten Abbildung der Topologie in den Merkmalsraum, sind die Ergebnisse zwiespältig. Während die Anordnung zwischen den Systemzuständen lokal sehr gut aus den Observationen abgeleitet wird und zu passenden Anordnungen führt, weist der erlernte Merkmalsraum in Gänze auch einige Faltungen und Deformationen auf. Das Training auf zusätzliche, in einem Experiment verfügbare Informationen, z.b. teilweise bekannte Systemzustände der Observationen, bietet an dieser Stelle Potenzial.

2. Ungleichmäßige Verteilung der Daten Problematisch erweist sich aber das Training der Autoencoder auf kleinen und ungleichmäßigen Trainingsmengen (siehe Tabelle 5.5). Zwar sind 3 100 Trainingsbilder (4 Bilder je möglicher Agentenposition) vergleichsweise wenig Trainingsdaten gegenüber den 50 000 Trainingsbildern in MNIST (5 000 Bilder je Klasse), so gesehen ist erstaunlich, dass die Autoencoder gute Ergebnisse erzielen. Allerdings müssen bereits bei dieser Menge rezeptive Felder eingesetzt werden, da die vollvernetzten Autoenocder mit noch weit mehr Gewichten bereits Schwächen bei der Generalisierung zeigen und deutlich schlechtere Rekonstruktionen der Testbilder erzeugen. Wird die Menge der Beobachtungen bis deutlich unter eine 100%-ige Abdeckung abgesenkt, werden die Rekonstruktions- und Klassifikationsergebnisse schnell schlechter und es kommt zu einer großen Zahl von "Verwechselungen", in denen Merkmalsvektoren auf falsche Positionen abgebildet werden und Rekonstruktionen den Agenten an einer falschen Stelle zeigen. Bereits in Kapitel 1, der Einleitung, wurde die klare Forderung formuliert, dass der Agent in der Lage sein muss, auf neue, bisher nicht gesehene Bilder zu generalisieren und nicht nur Observationen auswendig lernen darf.

An dieser Stelle bietet der Einsatz von Faltungskernen einen Ausweg. Faltungskerne können insbesondere dann eingesetzt werden, wenn sich, wie in den hier untersuchten Navigationsproblemen mit globaler Kamera, ein Objekt im Bild bewegt,

dessen Aussehen gleich bleibt. In diesem Fall ist es nicht nötig, lokale rezeptive Felder mit individuellen Gewichten zu trainieren; das Objekt kann überall im Bild auf die gleiche Weise mit den gleichen Gewichten entdeckt werden.

Aufgrund der geringeren Anzahl der Gewichte und des ortsunabhängigen Trainings führt der Einsatz der Faltungskerne zu deutlichen Verbesserungen auf kleinen Datensätzen. Selbst auf nur 155 Bildern, die im Grid-World-Experiment typischerweise innerhalb der ersten 10 Episoden eingesammelt werden, erlernen die Autoencoder mit Faltungskernen durchaus brauchbare Einbettungen, mit denen es bereits gelingt, mehr als 60% der Testbilder (siehe Tabelle 5.5, 9×9-Kern, fixierte Gewichte) richtig zu klassifizieren – obwohl noch nicht einmal 20% der Positionen besucht wurden.

Die Ergebnisse aus den Untersuchungen auf den synthetischen Bildern bestätigen sich voll bei der Anwendung auf die Bilder der Carrerabahn. Die Position des Wagens wird zuverlässig von einem Encoder mit Faltungskernen und ca. 800 000 Verbindungen extrahiert und kodiert. Im Merkmalsraum ist der Streckenverlauf der Bahn gut zu erkennen. Kleine Probleme entstehen mit wenigen Überschneidungen und Sprüngen im Merkmalsraum, die zu Verwechselungen führen können.

Im Hinblick auf die angestrebte Anwendung der beschriebenen Techniken im Zusammenspiel mit RL kann die grundsätzliche Frage gestellt werden, ob der Umweg über einen Autoencoder und die Kodierung der Daten in einem Merkmalsraum wirklich nötig ist. Alternativ könnte auch gleich ein flaches Netz verwendet werden, das direkt auf die Abbildung von Bildern auf eindimensionale Zustandswerte trainiert wird. Gegen ein solches direktes Vorgehen sprechen allerdings zwei voneinander unabhängige Argumente:

- Generell hat sich in direkten Vergleichsexperimenten gezeigt, dass bei Klassifikationsaufgaben mit hochdimensionalen Eingaben der Umweg über eine tiefe Einbettung der Bilder in einem Merkmalsraum und anschließendes Erlernen der Klassifikation auf Basis der Einbettung in aller Regel zu besseren Ergebnissen bei der Generalisierung als ein direktes Training flacher Netze oder zufällig initialisierter tiefer Netze führt [38, 83, 168]. So nimmt der von Hinton entwickelte, auf einem tiefen Encoder basierende Klassifikator im MNIST-Ranking[1] mit einem Klassifikationsfehler von nur 1.2% einen der vorderen Plätze unter den nicht speziell auf den Datensatz zugeschnittenen Methoden ein und liegt noch vor dem besten Ergebnis einer Support-Vector-Machine mit 1.4% sowie deutlich vor dem besten flachen, zufällig initialisierten und direkt auf die Klassifikation trainierten neuronalen Netz mit 1.6% [61, 88].

  Diese Ergebnisse bestätigen sich auch in eigenen Vergleichsexperimenten für das Labyrinthproblem, und das umso stärker, je weniger Trainingsdaten zur Verfügung stehen. Im bereits beschriebenen Zustandswertfunktionsexperiment erzielt ein flaches Netz mit optimierter Anzahl versteckter Neuronen bei 100% Abdeckung (775

[1] http://yann.lecun.com/exdb/mnist/, besucht im Feb. 2010

Trainingsbilder) eine Klassifikationsrate von 67.65% auf den Testbildern (Early stopping nötig, 100% auf den Trainingsbildern können erreicht werden). Auch mit einem mittels RProp trainierten Netz mit zwei versteckten Schichten konnte im besten Fall nur eine Klassifikationsrate von 70.8% erreicht werden, während die tiefen Netze im gleichen Experiment deutlich bessere 95.4% erzielen. Sind wesentlich mehr Trainingsdaten vorhanden, verschwindet der Unterschied (siehe Tab. 5.7).

Anscheinend verhindert der Umweg über die nichtlineare Einbettung sehr effektiv ein Überspezialisieren auf einzelne Pixel und führt so zum Erreichen eines hinsichtlich der Generalisierung besseren lokalen Minimums.

- Ein weiteres Argument hängt direkt mit dem Einsatz im Reinforcement Lernen zusammen: Anfangs ist über die tatsächliche (optimale) Wertfunktion nichts bekannt und es stehen folglich keine Zielwerte zur Verfügung, die indirekt Aufschluss über zusammengehörige und unterschiedliche Bilder geben könnten. Um jetzt eine bessere Annährung der Wertfunktion über einen Schritt des dynamischen Programmierens bestimmen zu können, ist aber die Unterscheidung einzelner Zustände nötig. In dieser Situation ist es hilfreich, dass sich die über einen Autoencoder gewonnene Einbettung der Daten unabhängig von der Approximation der Wertfunktion entwickelt und schon in dieser frühen Phase allein durch das Einsammeln von weiteren, sich unterscheidenden Beobachtungen verbessert. Die so völlig selbstorganisiert entstehende Unterscheidungsmöglichkeit der Bilder im Merkmalsraum kann dann als Basis für eine stabilere Schätzung der optimalen Wertfunktion verwendet werden.

Tabelle 5.7: Vergleich der Generalisierungsleistung im Zustandswert-Klassifikations-Experiment bei Verwendung eines tiefen, zunächst auf die Einbettung trainierten Netzes und bei Verwendung eines flachen, direkt auf die Zustandswertfunktion trainierten Netzes.

Netztyp	0.2	1	4
Flach, eine versteckte Schicht	58.06%	67.65%	99.11%
Flach, zwei versteckte Schichten	58.05%	70.81%	99.97%
Tiefer Encoder + Tasknetz	74.19%	95.43%	99.97%

Abschließend sei noch eine wichtige Bemerkung hinsichtlich der weiteren Verwendung der erzeugten Merkmalsräume in DFQ gemacht: Aufgrund des selbstorganisierten Erlernens der Einbettung ist vorab nicht bekannt, wie die Lage der einzelnen Bilder im Merkmalsraum genau sein wird. Jeder neue Lernvorgang führt zu gänzlich anderen Anordnungen. Auch wenn die Trennbarkeit überall im Merkmalsraum gegeben ist, kann die "Auflösung" lokal stark variieren; es gibt gedehnte Cluster, die einen relativ großen Bereich einnehmen und gleichzeitig Cluster, die sehr eng und feingliedrig beieinander liegen (vgl. Abbildung 5.5). Bei der Carrerabahn werden die Observationen gar auf eine

Mannigfaltigkeit des Merkmalsraums in Form eines schmalen Bandes abgebildet. In diesem Fall wird ein großer Teil des Merkmalsraums gar nicht genutzt, entlang des Bandes wird aber eine hohe Auflösung benötigt.

An den im optimierenden Lernen eingesetzten Approximator für die Wertfunktion werden daher besondere Anforderungen gestellt. Der Umgang mit ihnen wird im folgenden Kapitel diskutiert.

6

Adaptive Partitionierung des Merkmalsraums in DFQ

In Kapitel 4 wurden die theoretischen Anforderungen an die in DFQ zum Einsatz kommenden Funktionsapproximatoren erarbeitet, wenn Stabilität gewährleistet werden soll. In Kapitel 5 wurde auf die besonderen Bedingungen in Verbindung mit automatisch erzeugten Merkmalsräumen hingewiesen, die eine automatische Anpassung des Approximators an die Lage und Dichte der Daten im Merkmalsraum erfordern. In diesem Kapitel werden nun die Auswirkungen dieser Forderungen auf die Auswahl eines Funktionsapproximators diskutiert. In Ermangelung eines geeigneten Verfahrens wird mit dem ClusterRL-Algorithmus eine Variante von FQI mit irregulären Gitterapproximatoren entwickelt, die den mit der Stabilität und der automatischen Anpassung an die Lage der Daten verbundenen Anforderungen genügt. Eine Validierung des Verfahrens und ein Vergleich zur Verwendung regulärer Gitter wird auf einem häufig behandelten Standard-RL-Benchmark vorgenommen, bevor es in Abschnitt 6.5 in den DFQ-Algorithmus integriert und im Kapitel 7 auf visuomotorischen Lernproblemen zum Einsatz kommt.

6.1 Motivation

Dem innerhalb von DFQ zur Approximation der Zustandswertfunktion eingesetzten Funktionsapproximationsschema kommt eine entscheidende Bedeutung zu. Von seinen Eigenschaften wie der Ausdrucksstärke und der Generalisierungsfähigkeit hängt ganz entscheidend die Qualität der mittels DFQ erlernten Wertfunktionen und Strategien ab. Hinzu kommen die spezifischen Forderungen nach der Eignung für die automatisch erzeugten Merkmalsräume und nach der Stabilität innerhalb von DFQ. Da, wie in Kapitel 4 gezeigt, nur Averager zu einer stabilen Konvergenz in DFQ führen können, scheiden durch die letzte Forderung bereits eine Vielzahl von eigentlich interessanten Methoden inklusive linearer Regression, Multigrids (CMAC), RBF-Netze und MLPs aus. Gerade die MLPs böten sich eigentlich für eine Verwendung innerhalb von DFQ an, da sie sich dem Merkmalsraum beliebig anpassen und ihn potenziell sogar über ein Zurückpropagieren der Fehlergradienten verändern könnten (siehe Abschnitt 5.3.4). Außerdem

waren sie in der Praxis im Einsatz in batch RL-Verfahren (NFQ) bereits sehr erfolgreich [74, 130, 133, 134]. Aufgrund des komplexen Lernvorgangs innerhalb von DFQ und der mit den Generationswechseln bereits bestehenden Quelle für Schwankungen im Lernverlauf (siehe Abschnit 4.5.2) fiel die Entscheidung aber zugunsten der Beschränkung auf stabile Funktionsapproximationsverfahren in DFQ – nicht nur aus theoretischen, sondern auch aus ganz praktischen Gründen.

Reguläre Gitter Von den in Frage kommenden stabilen Averaging-Methoden sind die Gitterapproximatoren mit regulären Gitterstrukturen im Reinforcement Lernen wohl am weitesten verbreitet. Mit der Partitionierung des Zustandsraums in reguläre Hyperrechtecke stellen die regulären Gitterapproximatoren prinzipiell die einfachste Methode zur Diskretisierung des Zustandsraums dar. Zwischen den regulär angeordneten Vertices können gespeicherte Funkionswerte dann linear interpoliert werden (lineare Gitterapproximatoren) oder es kann für jede Hyperzelle ein konstanter Funktionswert verwendet werden (konstante Gitterapproximatoren). Gegen den Einsatz dieser konzeptionell einfachen Averager in DFQ spricht aber, dass ihre Leistung entscheidend von der passenden Einstellung der Gitterauflösung abhängt. Ist die Auflösung nicht hoch genug, sinkt die Approximationsgenauigkeit und im schlimmsten Fall kann es aufgrund von Zustandsverdeckungen zu einer generellen Inkompatibilität des Approximators mit dem MDP und damit zu Divergenz in kürzester Pfad Problemen kommen (siehe Kapitel 4). Wird die Auflösung dagegen zu fein eingesetellt, schadet es zwar nicht der Stabilität führt aber zu deutlichen Einbußen bei der Dateneffizienz, weil für die adäquate Approximation der Wertfunktion mehr Daten als beim optimalen Gitter benötigt werden (siehe auch eigene empirische Untersuchungen in Abschnitt 6.4).

Dieses Verhalten ist insofern problematisch, weil im Ablauf von DFQ die optimale Gittergröße nicht wie in anderen Lernversuchen durch eine Analyse des Zustandsraums manuell oder auch empirisch durch systematisches "Ausprobieren" in aufwändigen Versuchsreihen bestimmt werden kann. Die in DFQ verwendeten tiefen Autoencoder konvergieren zu einem lokalen Minimum des Rekonstruktionsfehlers, wobei die dabei entstehende Anordnung der Observationen im Merkmalsraum jedes Mal eine andere ist. Die Eigenschaften der Merkmalsräume, auf denen der Approximator arbeiten soll, sind daher vorab nicht bekannt, so dass die Gittergröße nicht vor Trainingsbeginn bestimmt, sondern automatisch an die erzeugten Merkmalsräume angepasst werden müsste. Außerdem sind auch lokal unterschiedliche Auflösungen für eine optimale Anpassung an die Lage der Daten nötig. Denn aufgrund der teilweise sehr unterschiedlichen Größen und Abstände einzelner Cluster gibt es keine Auflösung, die in allen Bereichen des Merkmalsraums einen optimalen Kompromiss zwischen Genauigkeit und Dateneffizienz bietet. Zur Approximation der Wertfunktion auf Basis der in einer Mannigfaltigkeit liegenden Merkmalsvektoren (Band im Carrerabahn-Beispiel in Abschnitt 5.3.7) werden beispielsweise viele Zellen entlang der Mannigfaltigkeit und eigentlich gar keine Zellen außerhalb der Struktur benötigt.

Iterative Verfeinerung von Gittern und Bäumen In der Literatur findet sich im Bereich des Reinforcement Lernens zu diesem Problem ein von Munos and Moore beschriebenes Gitterverfahren mit variabler Auflösung [103, 104]. Ausgehend von einem gewöhnlichen regulären Gitter werden die Hyperrechtecke nach bestimmten Kriterien dort wo nötig aufgeteilt und die in einem speziellen kd-Baum [75] repräsentierte Gitterstruktur so in einem iterativen Prozess nach und nach verfeinert. Die Idee ist es, nur dort viele Hyperzellen zu platzieren, wo eine hohe Genauigkeit vonnöten ist, und in anderen, uninteressanten Bereichen mit einer niedrigeren Auflösung zu arbeiten. Reynolds hat den Ansatz auf modellfreies online Reinforcement Lernen übertragen, mit Aufteilungskriterien, die lokal an den Zellen geführte Statistiken verwenden [125, 126]. Auch Ernsts Einsatz von randomisierten Bäumen innerhalb von FQI ist als eine zu diesen Verfahren verwandte Methode zu sehen [39].

Für die Aufteilung der Hyperzellen wurden verschiedene Kriterien entwickelt, die auf der Dichte der Daten basieren [39, 126], die den Unterschied in den Kostenschätzungen heranziehen [104] bzw. die Varianz in den durchgeführten Aktualisierungen schätzen [126] oder auch versuchen, komplexere Eigenschaften der Wertfunktion bzw. der optimalen Strategie zu analysieren [104]. Allerdings sind diese Verfahren im Einsatz nicht unproblematisch. Im Verfahren von Reynolds, das, weil modellfrei, für DFQ besonders interessant ist, gibt es in den Kriterien eine Reihe nicht einfach zu bestimmender Parameter. Sowohl in [104] als auch in [126] ist zudem zu sehen, dass die vorgeschlagenen Varianzkriterien vor allem zu einer fortwährenden Verfeinerung des Gitters an durch rechteckige Hyperzellen nicht exakt zu erfassenden, aber für die optimale Trajektorie unwichtigen Diskontinuitäten führen, während die Auflösung entlang der optimalen Trajektorie niedrig bleibt. Bei dem für modellfreies Lernen interessanten Verfahren von Reynolds fehlt zudem jegliches Kriterium dafür, wann die iterative Aufsplittung in der Praxis gestoppt werden kann. Allen drei Verfahren ist gemein, dass sie auf achsenparallele Zellgrenzen beschränkt sind. Selbst im Fall optimal vorgenommener Aufteilungen können schräg liegende Diskontinuitäten von ihnen nicht richtig erfasst werden und führen zu den beobachteten, fortwährenden Verfeinerungen.

Diese Verfahren können trotz aller Einschränkungen helfen, eine anfangs reguläre Gitterstruktur an die Erfordernisse eines automatisch erzeugten Merkmalsraums anzupassen. In der Anwendung ist Reynolds für modellfreies Lernen prinzipiell geeignete Methode aber nicht ganz einfach zu handhaben und kommt auch nicht ohne die Einstellung problemspezifischer Parameter aus.

Irreguläre Gitter Sind wie im batch RL alle für die Approximation verwendeten Daten von Anfang an vorhanden, gibt es keinen Grund mehr, den Lernvorgang zunächst mit einem gewöhnlichen regulären Gitterapproximator zu starten und diesen dann aufwändig nach und nach über die Fortführung lokaler Statistiken zu verfeinern. Im batch RL gibt es auch für die Beschränkung der Form der Gitterzellen auf achsenparallele Hyperrechtecke keine weitergehende theoretische Notwendigkeit: Die vorhandenen Konvergenzaussagen gelten für völlig irreguläre Gitter mit beliebig orientierten Zellgrenzen genauso wie für achsenparallele, reguläre Gitter [53, 97, 113]. Zwar ist

der Einsatz völlig irregulärer Gitter im Reinforcement Lernen eher ungewöhnlich. Bei der näherungsweisen Lösung von gewöhnlichen und partiellen Differentialgleichungen (Finite-Elemente-Methode, Finite-Differenzen-Methode) sind diese irregulären Gitter aber völlig gebräuchlich [8, 110] und werden auch als eine Möglichkeit angesehen, den exponentiellen Anstieg der Komplexität mit der Dimension des Zustandsraums ("curse of dimensionality") gegenüber dem Einsatz regulärer Gitter abzumildern [13, 95]. Von Merke wurden die irregulären Gitterapproximatoren bereits explizit für den Einsatz in einem ähnlichen batch RL-Szenario vorgeschlagen und dort zwar nicht praktisch erprobt, aber theoretisch untersucht [97].

Natürlich kann auch bei den irregulären Gittern die Strukturanpassung in DFQ nicht per Hand vorgenommen werden. Da alle Daten von Beginn an vorliegen, kann die Struktur aber mittels gut studierter Methoden aus der Clusteranalyse automatisch auf die Lage und Verteilung der Daten im von DFQ erzeugten Merkmalsraum abgestimmt werden und bedarf keiner speziellen, ungewöhnlichen Algorithmen mehr. Statt mit einem groben Gitter zu beginnen und dieses aufwändig mit speziellen, schwer zu kontrollierenden Aufteilungskriterien iterativ zu verfeinern, werden die Daten im hier vorgestellten ClusterRL-Verfahren gleich in die Verteilung widerspiegelnde Cluster euklidisch benachbarter Mermalsvektoren eingeteilt. Auf den gefundenen Zentroiden bzw. Prototypen kann dann leicht eine Gitterstruktur aufgespannt werden. So lässt sich mittels der Delaunay-Triangulation [30] eine aus Dreiecken bestehende Gitterstruktur finden, in der Funktionswerte an den auf die Vertices fallenden Stützstellen gespeichert und in den Zwischenräumen zum Beispiel mit Hilfe baryzentrischer Interpolation linear interpoliert werden können. Mit Hilfe des Voronoi-Diagramms lassen sich Gitter finden, in denen alle Datenpunkte, die demselben Zentroiden zugeordnet werden, den gleichen, konstanten Funktionswert zugewiesen bekommen. Delaunay-Triangulation und Voronoi-Diagramm sind duale Graphen – die eine Darstellung kann aus der anderen berechnet werden – und sind in Abbildung 6.1 für zwei unterschiedliche Anordnungen von Vertices dargestellt. In diesem Sinne können auch die regulären Gitter als ein Spezialfall der auf Zentroiden aufgespannten Gitterstrukturen interpretiert werden. Linear interpolierende Gitterapproximatoren entsprechen einer Delaunay-Triangulation auf den regulär angeordneten Zentroiden mit baryzentrischer Interpolation zwischen den Vertices, konstante Gitterapproximatoren entsprechen dem Voronoi-Diagramm mit einem konstanten, am jeweiligen Zentroiden gespeicherten Funktionswert für alle Punkte innerhalb der gleichen Zelle.

Gegenüber anderen Verfahren bieten die irregulären Gitter den Vorteil, dass sich ihre Struktur auf sehr einfache Weise mit gut studierten Verfahren an die lokalen Anforderungen anpassen lässt, dass sie sich ansonsten aber genauso stabil und einfach wie reguläre Gitterapproximatoren innerhalb von batch RL-Methoden verwenden lassen. Gleichzeitig fällt bei ihrer Verwendung die Beschränkung auf Hyperrechtecke und achsenparallele Zellgrenzen weg.

Kernelbasierte, nicht-parametrische Methoden Eine Anpassung der Lage und Dichte der Stützstellen eines Funktionsapproximators ist nur eine Möglichkeit, die Schät-

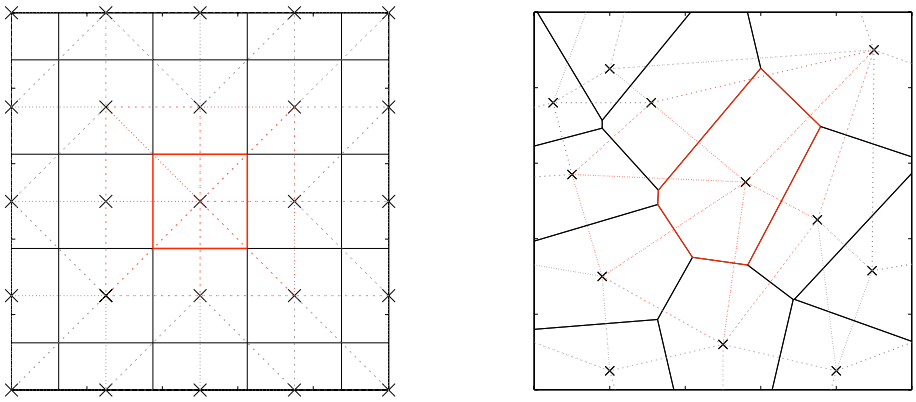


Abbildung 6.1: Duale Graphen Voronoi-Diagramm (durchgezogene Linien) und Delaunay-Triangulation (gestrichelte Linien) zu vorgegebenen Zentroiden (Kreuze) in einer regulären Anordnung (links) und in irregulärer Anordnung (rechts).

zung an die Dichte der Daten anzupassen. Stattdessen könnten auch nicht-parametrische kernelbasierte Methoden wie n-Nächste-Nachbarn, Case Based Reasoning (CBR) oder Kernel-Averaging entsprechend Definition 6 (in Kapitel 4) verwendet werden. Diese Verfahren würden für die Schätzung ganz automatisch in Gebieten hoher Dichte sehr nahe liegende Daten verwenden bzw. höher gewichten und in Gebieten niedrigerer Datendichte auch auf weiter entfernt liegende Datenpunkte zurückgreifen. Eine solche, nicht-parametrische Approximation liegt näher an der ursprünglichen Idee von KADP [113], während die Verwendung von parametrischen-Verfahren mit Quantisierung, Prototypen und explizit repräsentierten Approximationen wie im hier vorgeschlagenen ClusterRL-Verfahren näher an FVI [53] und FQI [39] liegt.

Ohne sich auf eine dogmatische Diskussion einzulassen – es gibt für beide Möglichkeiten nicht zu widerlegende Argumente – wurde im Rahmen dieser Arbeit aus folgenden Gründen eine pragmatische Entscheidung für die explizit repräsentierten Gitterapproximatoren getroffen:

- In den parameterfreien Methoden werden auch in der Anwendungsphase alle Transitionen benötigt, während die Gitter nur die Werte an den dünner gesäten Stützstellen benötigen.

- Aufgrund des Zugriffs auf alle Daten und der für jede Anfrage neu durchgeführten Mittelung ("lazy-learning") sind die nicht-parametrischen Verfahren in der Anwendungsphase langsamer als die Gitter, was bei großen Datenmengen unter Umständen die Echtzeitfähigkeit beeinträchtigt.

- Ein möglicher Vorteil der nicht-parametrischen Verfahren ist die durchgeführte Glättung in Abhängigkeit von der Distanz zur jeweiligen Anfrage, ohne harten Grenzen im Zustandsraum zu ziehen. In der praktischen Anwendung in DFQ und

im RL allgemein ist diese Eigenschaft aber nicht unbedingt ein Vorteil.

In vielen Reinforcement Lernproblemen treten häufig harte Entscheidungsgrenzen und Sprünge in der Wertfunktion auf, gerade bei der Verwendung von diskreten Zeitschritten und der Modellierung als kürzester Pfad Problem. Das Erfassen dieser harten Grenzen stellt bei der Approximation das größte Problem dar, nicht die Approximation von weichen, fließenden Übergängen Lipschitz-stetiger Funktionen, wie sie aus theoretischer Sicht in Kapitel 4 diskutiert wurde. Auch wenn dieser Aspekt in der vorliegenden Arbeit nur eine untergeordnete Rolle spielt, bieten prototypbasierte Verfahren immerhin potenziell die Möglichkeit, harte Grenzen entlang dieser Diskontinuitäten durch den Zustandsraum zu ziehen – während die nicht-parametrischen Methoden Datenpunkte auf der richtigen und falschen Seite der Diskontinuität einbeziehen.

In den folgenden Abschnitten werden zunächst die verschiedenen Möglichkeiten der Gitteroptimierung und dann das neu entwickelte ClusterRL-Verfahren beschrieben. In Abschnitt 6.4 erfolgt eine Evaluation der irregulären Gitterapproximatoren im Einsatz in FQI. In Abschnitt 6.5 wird dann die Integration des ClusterRL-Verfahrens in DFQ diskutiert und die Variante DFQ-C (DFQ mit Clustern) vorgestellt.

6.2 Optimierung der Struktur eines Gitterapproximators

Die Idee, Methoden aus der Clusteranalyse und dem prototypbasierten Lernen auf den Zustandsraum in RL anzuwenden und die Wertfunktion an den Zentroiden der Cluster zu approximieren, ist nicht neu und bildete bereits die Grundlage für eine Reihe von Publikationen[1] unter Verwendung von Vektorquantisierung (VQ) und fuzzy-VQ [41, 90], Neural Gas (NG) [56] und Self-Organizing Maps (SOM) [35, 36, 151, 167]. Ein interessanter Ansatz ist die Arbeit von Tluk von Toschanowitz, in der Zustands-Aktionspaare nicht wie bei den anderen Arbeiten im Zustandsraum, sondern auf Basis der Q-Werte mittels einer NG-ähnlichen Lernregel zusammengelegt werden [165, 166]. Für diskrete Zustandsräume und online Aktualisierungen finden sich die Grundzüge der Idee, den Zustandsraum zu clustern, zusammen mit einer theoretischen Analyse unter dem Namen "adaptive soft state aggregation" bereits in [149]. Dort wird in einem vorläufigen ("preliminary") empirischen Versuchsergebnis auch das grundsätzliche Potenzial dieser Verfahren angedeutet.

Beim batch Reinforcement Lernen auf einer Stichprobe stehen bereits alle Daten zu Beginn zur Verfügung und während des Lernvorgangs kommen keine neue Transitionen hinzu. Es könnten also auch bei der Optimierung der Lage der Stützstellen passende batch Verfahren aus der Clusteranalyse zum Einsatz kommen. Erstaunlicherweise handelt es sich bei den oben aufgezählten Verfahren aber ausnahmslos um Methoden mit online DP-Aktualisierungen und online Verfahren zur Vektorquantisierung, während

[1] Erstaunlich ist allerdings, dass diese eigentlich sehr nahe liegende und attraktive Idee zwar schon untersucht wurde, die Anzahl der Publikationen und erfolgreichen Anwendungen aber sehr klein ist. Insgesamt spielen diese Verfahren nur eine sehr untergeordnete Rolle, wobei die SOMs noch am häufigsten zu finden ist, während andere Approximatoren – CMACs, RBFs, MLPs und selbst reguläre Gitter – viel größere Aufmerksamkeit innerhalb der RL-Community erhalten.

eine Literaturrecherche nach der Kombination der zugehörigen batch Verfahren ergebnislos bleibt. Immerhin ein Ansatz wurde gefunden [120, 121], bei dem zwar keine batch Verfahren verwendet werden, aber die Optimierung der Stützstellen und das online Strategielernen zeitlich separiert bleiben: Zuerst wird mittels Growing Neural Gas (GNG) [44] die Lage von Prototpyen auf Basis der mittels einer Zufallsstrategie observierten Zustände optimiert. Erst anschließend wird auf Basis der gefundenen Prototypen mittels SARSA (siehe [156]) eine einfache Navigationsstrategie für einen simulierten Roboter erlernt. Aufgrund der eingesetzten komplexen Optionen und des Reward-Shapings ist an Hand der Auswertungen schwer einzuschätzen, wie gut das Verfahren im allgemeinen Fall arbeitet. In [120] wird als Ziel der weiteren Arbeiten die Kombination von Aktualisierungen der Prototypen des GNGs und des Strategielernens benannt. Die "geringe" Zahl der Funde für die Kombination von batch RL mit Methoden aus der Clusteranalyse mag damit zu erklären sein, dass die batch RL-Verfahren selbst noch relativ jung, und nicht so weit verbreitet wie online Q-Lernen oder SARSA sind.

In Hinblick auf den Einsatz in DFQ ist die Wahl zwischen batch und online Verfahren der Vektorquantisierung nicht beliebig. Es gibt zwei voneinander unabhängige Probleme beim beim Einsatz von online VQ-Methoden zur Lagebestimmung von Prototypen im Reinforcement Lernen. So verleiten die Aktualisierungsregeln der NG- und SOM-Algorithmen dazu, nicht nur den Wert der nahe gelegensten Stützstelle zu aktualiseren, sondern auch an weiteren Stützstellen gespeicherte Q-Werte – gewichtet entsprechend einer Nachbarschafts- oder Zugehörigkeitsfunktion – Richtung der aktuellen Schätzung zu bewegen [35, 36, 56, 151, 167]. Grundsätzlich ist ein solches Vorgehen auch im Einsatz mit batch RL (konkret FQI) stabil möglich [53], aber es muss sichergestellt werden, dass die Bedingungen (4.34) und (4.35) für alle Gewichte eingehalten werden. Mindestens eine Normalisierung der Gewichtung über alle Datenpunkte entsprechend der Konstruktion (4.36) ist nötig. Prinzipiell muss also für jeden Einzelfall geprüft werden, ob sich die konkrete Lernregel als kernelbasierter Averager repräsentieren lässt und die vorgenommene Approximation somit nicht-expansiv und mit den batch RL-Verfahren verträglich ist. Merke liefert ein konkretes Beispiel dafür, wie leicht eine intuitiv konstruierte und völlig ungefährlich erscheinende Lernregel – die gleichzeitige Aktualisierung mehrerer Stützstellen in einem linear interpolierenden Gitterapproximator entsprechend der Gewichtung mittels der baryzentrischen Koordinaten des Trainingsbeispiels [97] – auch für $\lambda < 1$ zu Divergenz führen kann. Ein weiteres Beispiel für eine unproblematisch erscheinende Aktualisierung mehrerer Stützstellen entsprechend einer Zugehörigkeitsfunktion ist das CMAC, das bei näherer Betrachtung kein Averager ist, und von Gordon [53] und auch Merke [97] explizit als Beispiel für instabiles, divergierendes Verhalten in batch RL besprochen wird. Ganz von den Stabilitätsproblemen abgesehen, ist grundsätzlich der Sinn fraglich, weiter entfernt liegende Stützstellen in Richtung des Q-Wertes des aktuellen Trainingsbeispiels zu ziehen. Zumindest in zeitdiskreten kürzester Pfad Problemen und überall dort, wo die Wertfunktion nicht Lipschitz-stetig ist, sondern Diskontinuitäten Probleme bereiten, sollte eine solche Aktualisierungsregel vorsichtig eingesetzt werden.

Ein weiteres Problem entsteht durch den Wechsel zwischen Verschiebung der Stütz-

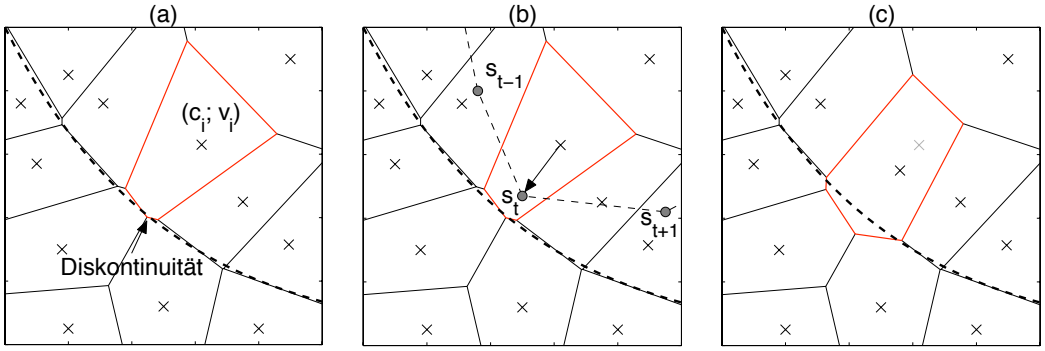


Abbildung 6.2: Problematik bei der Verwendung von VQ. Eine Beobachtung s_t zieht den Prototypen c_i in seine Richtung (b), so dass sich sein Einflussbereich ändert. Durch die Verschiebung ändert sich der tatsächliche Erwartungswert $E[R_t|s_t \in C_i]$ im Einflussbereich, wodurch die gespeicherte Schätzung v_i praktisch schlechter wird. Besonders dramatisch fällt der entstehende Fehler im Bereich von Diskontinuitäten aus. Im Beispiel ragt der Einflussbereich anders als vorher (a) nach der Aktualisierung über die Diskontinuität hinaus (c). Das kann selbst dann passieren, wenn weder die fragliche Beobachtung noch ein Teil des Einflussbereiches auf der anderen Seite der Diskontinuität liegen.

stellen und Aktualisierung der an ihnen gespeicherten Zustands- oder Q-Werten. Die Schätzung der Kosten an einer Stützstelle hängt von den Übergängen ab, die in ihrem Einzugsbereich liegen und für die Aktualisierung der Kostenschätzung verwendet werden – sei es durch Q-Lernen oder sei es durch FQI. Wird die Stützstelle bewegt, ändern sich der Einzugsbereich und damit auch die zu erwartenden Kosten. Bezeichne C_i zum Beispiel die Menge aller Zustände $s \in S$, die in den Einzugsbereich des Prototypen c_i in einem konstanten, irregulären Gitter fallen (siehe Abbildung 6.2). Das Ziel eines Reinforcement Lerners könnte es nun sein, in dem an der Stützstelle c_i gespeicherten Wert v_i die zu erwartenden Kosten $E_{\pi*}[R_t|\ s_t \in C_i]$ zu schätzen, wenn sich der Agent zum Zeitpunkt t "innerhalb" von C_i befindet und ab dann die optimale Strategie verwendet. Wird nun der Zentroid bewegt, ergibt sich das Problem, dass sich der Einzugsbereich ändert und der neue Einzugsbereich C_i' an der neuen Stelle c_i' nicht dem alten entspricht, also $C_i' \neq C_i$. Daher ändert sich auch der Erwartungswert, so dass wahrscheinlich $E_{\pi*}[R_t|\ s_t \in C_i'] \neq E_{\pi*}[R_t|\ s_t \in C_i]$. Die Änderung fällt für kleine Bewegungen und glatte optimale Q-Funktionen unter Umständen nur klein aus, kann aber aufgrund von Diskontinuitäten in der Wertfunktion selbst für kleine Änderungen schnell anwachsen. Ohne weitere Annahmen über die Struktur der optimalen Wertfunktion, die Größe der Veränderungen der Stützstellen und die Lage der Daten ist die bisherige Schätzung v_i hinfällig und nicht besser als jeder andere initiale Wert an dieser Stützstelle.

Ein abstrakteres Beispiel ist in Abbildung 6.3 zu sehen. Wird die Stützstelle x_2, an der der Funktionswert $f(x_2)$ (Schätzung im Beispiel blau markiert) vom linear zwischen den Stützstellen interpolierenden Averager bereits genau approximiert wurde, nach links verschoben, verschlechtert sich offensichtlich die approximierte Funktion f'

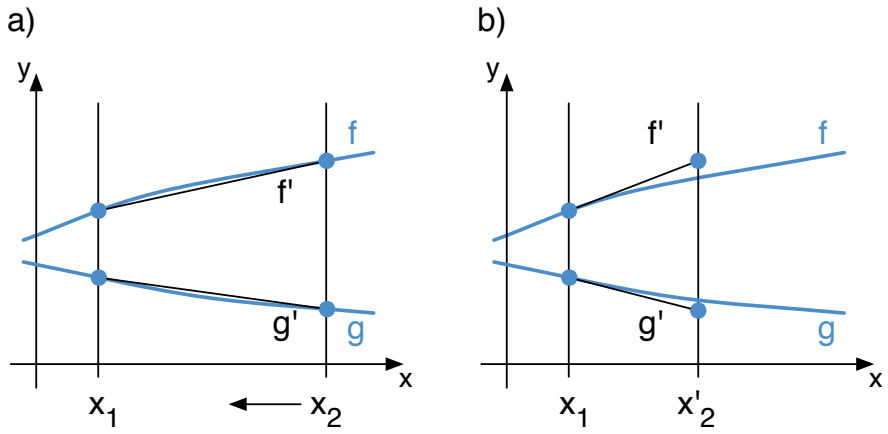


Abbildung 6.3: Verschlechterung der Approximation durch Verschiebung der Stützstellen. a) Approximation zweier Funktionen f und g an den Stützstellen x_1 und x_2 und approximierte Funktionen f' und g' (schwarze Linien). b) Situation nach einer Verschiebung der Stützstelle x_2 an die Stelle x'_2. Beschreibung siehe Text.

(siehe Bild b). Im Falle von inkrementellen Aktualisierungen der Funktionswerte (z.B. online Q-Lernen) sind mehrere Aktualisierungen nötig, um diesen, durch die eine Veränderung der Position der Stützstelle hervorgerufenen Fehler $f(x_2) - f(x'_2)$ zu korrigieren. Beim Einsatz von batch Verfahren wie FVI und KADP können diese Verschiebungen im Wechsel mit DP-Aktualisierungen zu schwerwiegenderen Problemen bis hin zur Divergenz führen. Abgesehen davon, dass sich die Stützstellen – und damit die Gewichte der Trainingsbeispiele – während des dynamischen Programmierens für ein konvergentes Verhalten nicht ändern dürfen (siehe Kapitel 4), ist hinsichtlich der Definition der nichtexpansiven Averager bei Gordon [53] auch kritisch, dass im Beispiel die approximierten Werte an der Stützstelle im Bild b) plötzlich weiter auseinander liegen als die zwei zu approximierenden Funktionen. Die reinen online Verfahren [35, 36, 56, 90, 151, 167], die eine Anpassung der Position der Stützstellen und eine Aktualisierung der an ihnen gespeicherten Q-Werte abwechselnd vornehmen, sind alle von dieser Problematik betroffen.

Aus diesen Gründen ist der Einsatz von reinen batch Verfahren und eine Entkopplung von Sturkuranpassung und Wertfunktionsapproximation vorzuziehen. Im Cluster-RL-Verfahren werden die Optimierung des Gitters und die DP-Iterationen daher gänzlich auseinander gehalten; erst werden die Stützstellen bestimmt, dann wird auf ihnen dynamisch programmiert. So wird ein stabiles Verhalten und die sichere Konvergenz des FQI-Verfahrens gewährleistet. Tatsächlich kann diese Separierung aber nur dateneffizient realisiert werden, wenn batch RL angewendet wird, und deshalb alle Transitionen vorab bekannt sind und beiden Verfahren zur Verfügung stehen.

6.3 ClusterRL-Algorithmus

Im ClusterRL-Algorithmus wird vor der eigentlichen Ausführung von Fitted Q-Iteration zunächst ein irreguläres Gitter auf die Daten angepasst. Gegeben einen Datensatz $\mathcal{F}$ von Übergängen in Form von p Quadrupeln $\mathcal{F} = \{(s_t, a_t, r_{t+1}, s_{t+1}) | t = 1, \ldots, p\}$, läuft der Algorithmus entsprechend der Beschreibung zu folgender Funktion ab:

Funktion `ClusterRL: Fitted Q-Iterations mit irregulärem Gitter`

Input : $k_{\text{init}}, min, \bar{q}^0, \mathcal{F}$
Result : Gitterapproximation der optimalen Wertfunktion $\bar{Q}$

$i \leftarrow 0$;
$X \leftarrow$ `extrahiere Zustände(`$\mathcal{F}$`)`;
$C_0 \leftarrow$ `initialisiere Zentroiden(`X,k_{init},$\bar{q}^0$`)`;
repeat
  $i \leftarrow i + 1$;
  $C_i \leftarrow$ `k-means(`C_{i-1}, X`)`;
  $C_i \leftarrow$ `entferne überflüssige Zentroiden(`C_i, X, min`)`;
until $|C_{i-1}| - |\mathcal{C}_i| = 0$;
$\hat{Q}^0 \leftarrow$ `erzeuge Gitterapproximator(`C_i, $\bar{q}^0$`)`;
$\bar{Q} \leftarrow$ `Fitted Q-Iterations(`$\hat{Q}^0, \mathcal{F}$`)`;

Zunächst wird die Funktion **initialisiere Zentroiden** mit der Menge X aller in $\mathcal{F}$ enthaltenen Ausgangszustände aufgerufen, um eine Menge $C_0 = \{c_i \,|\, i = 1, \ldots, m\}$ von $m = \min(p, k_{\text{init}})$ Prototypen c_i zu erzeugen. Die Positionen dieser Prototypen werden in einem Urnenexperiment ohne Zurücklegen aus den Datenpunkten ausgewählt.

Initialisierungsvariante für kürzester Pfad Probleme: Um zu verhindern, dass zu wenige Prototypen in der Nähe der als Endpunkte der Trajektorien seltenen, aber für das RL besonders wichtigen Übergänge in absorbierende Terminalzustände liegen, können zusätzliche Prototypen auf der Position der beobachteten Übergänge in absorbierende Terminalzustände initialisiert werden, sofern diese Übergänge noch nicht in C_0 enthalten sind. Dieses Vorgehen verhindert effektiv die Verdeckung der absorbierenden Terminalzustände – und anderer wichtiger Zustände – bei der Verwendung zu kleiner Gitter und hilft somit, die Schätzung der Kosten an den einmal besuchten Terminalzuständen zu "verankern". Darin ähnelt das Vorgehen der "Hint-to-Goal" Heuristik [130], mit dem Unterschied, dass hier kein Vorwissen benötigt, sondern die Prototypen automatisch nach dem Erreichen der Terminalzustände platziert werden.

Diese anfängliche Verteilung der Prototypen wird mittels eines Clusteranalyseverfahrens optimiert. Im Rahmen dieser Arbeit wird dazu das k-means Verfahren verwendet, das im folgenden Abschnitt 6.3.1 beschrieben wird. Es kann gegen andere Verfahren wie (Growing) Neural Gas [44, 94] und SOM – inklusive entsprechender batch Varianten [28, 57, 77] – ausgetauscht werden.

Variante mit Ausdünnung des initialen Gitters Optional wird nach Abschluss des k-means Verfahrens das Gitter ausgedünnt, indem alle Zentroiden gelöscht werden, in deren Einflussbereich zu wenige Daten für eine gute Schätzung der Kostenfunktion liegen. Aufgerufen wird die entsprechende Funktion **entferne überflüssige Zentroiden** mit den aktuellen Prototypen, den Datenpunkten und einem Parameter *min*, der die minimal erforderliche Anzahl Datenpunkte im Einflussbereich jedes Prototypen angibt. Bei jedem Aufruf der Methode wird jeweils nur maximal ein Prototyp entfernt, der die Bedingung nicht erfüllt. Dabei wird immer derjenige Prototyp entfernt, der die wenigsten Datenpunkte repräsentiert. Anschließend wird in einem weiteren Schleifendurchlauf die Gitterstruktur durch neuerlichen Aufruf der k-means Methode "repariert" und ein weiterer Prototyp entfernt, solange, bis keine Prototypen mehr vorhanden sind, die nicht mindestens *min* Datenpunkte im Einflussbereich haben. Diese Variante behebt nebenbei das Problem mit Prototypen ohne Daten innerhalb ihres Einzugsbereichs [28], das bei k-means im Falle sehr ungünstiger Aktualisierungen auftreten kann. Der Parameter *min* bietet prinzipiell eine weitere Möglichkeit zur Optimierung, wurde im Rahmen dieser Arbeit aber grundsätzlich auf die Anzahl der zur Verfügung stehenden Aktionen gesetzt, was durchweg zu guten Ergebnissen geführt hat.

Aus den so erzeugten Prototypen wird dann ein Gitterapproximator erzeugt, indem jedem Prototypen $c \in C$ neben seiner Position im Zustandsraum ein Vektor $q \in \mathbb{R}^{|A|}$ von Q-Werten q_i für jede der diskreten Aktionen $a_i \in A$ zugewiesen wird. Anfänglich werden diese Q-Werte auf den Wert des Parameters $\bar{q}^0$ gesetzt. Nun wird der Algorithmus Fitted Q-Iteration (siehe Abschnitt 2.2.8) aufgerufen, wobei in FQI direkt mit Schritt 2 begonnen wird, da die Initialisierung bereits hier vorgenommen wurde.

Die Durchführung von k-mean, das Abspeichern der Schätzung der Wertfunktion im Gitterapproximator (in FQI Schritt 3, überwachtes Lernen) und die vorgenommene Approximation zwischen den Stützstellen des Gitterapproximators werden in den folgenden drei Abschnitten beschrieben.

6.3.1 Lloyds Algorithmus (alias k-means)

In der k-means Formulierung des Clusteringproblems [154] wird unter einer Partitionierung der $N = |X|$ Eingabevektoren $X \subset \mathbb{R}^n$ in k echte Teilmengen (Cluster) $\bar{X}_1, \ldots, \bar{X}_k \subset X$ jeder zu einem Cluster $\bar{X}_i$ gehörende Datenpunkt $x_j \in \bar{X}_i$ einem gemeinsamen Zentroiden (oder auch Prototyp) $c_i \in \mathbb{R}^n$ zugewiesen. Gesucht werden nun k Clusterzentren $c_1, \ldots, c_k$ und eine Partitionierung der Datenpunkte, die den "globalen" Fehler, hier den quadrierten Abstand der Datenpunkte zu ihren zugeordneten Clusterzentren

$$D_C := \sum_{i=1}^{k} \sum_{j=1}^{N} ||c_i - x_j||^2 \cdot \delta_i(j)$$

minimieren, wobei C die Menge aller k Clusterzentren und δ_i eine Indikatorfunktion mit $\delta_i(j) \in \{0, 1\}$ für die Clusterzugehörigkeit zum Cluster $\bar{X}_i$ sei. Eine wichtige Be-

obachtung ist, dass unter der optimalen Lösung das Clusterzentrum c_i genau auf den Mittelpunkt

$$\mu_i = \frac{\sum_j x_j \delta_i(j)}{\sum_j \delta_i(j)}$$

der ihm zugeordneten Datenpunkte (für alle anderen $x_j \delta_i(j) = 0$) fallen muss, denn Ableiten der Kostenfunktion D_C nach den Clusterzentren c_i und anschließendes Nullsetzen führt zu dem Ergebnis, dass die Nullstelle und damit das Minimum von D_C genau bei $c_i = \mu_i$ liegt. Desweiteren muss bei einer optimalen Lösung gleichzeitig jeder Datenpunkt x genau dem Cluster $\bar{X}_i$ zugeordnet sein, für den der Abstand $||c_i - x||$ zum zugehörigen Clusterzentrum c_i minimal ist, denn wenn es ein anderes Clusterzentrum c_j mit $i \neq j$ und $||c_j - x|| < ||c_i - x||$ geben sollte, könnten die globalen Kosten D_C offensichtlich gesenkt werden, indem der Datenpunkt x dem Cluster $\bar{X}_j$ zugeordnet werden würde.

Der Algorithmus von Lloyd [92], auch bekannt als k-means Algorithmus, ist das verbreitetste Verfahren zur Lösung dieses Problems. Das Verfahren macht sich die beiden genannten Beobachtungen über die Gruppierung der Datenpunkte und die Position der Clusterzentren in der optimalen Lösung zunutze und setzt sie direkt in einen iterativen Algorithmus zur Lösung des Problems um:

Ausgehend von einer initialen Menge C^0 von k Clusterzentren wird über die folgenden beiden Schritte iteriert:

1. **Anpassung der Gruppen** Zuordnung jedes Datenpunkts $x \in X$ zu demjenigen Cluster $\bar{X}_i$, dessen zugehöriges Zentrum c_i dem Datenpunkt x am nächsten ist, also gilt $\forall j \ ||c_i - x|| \leq ||c_j - x||$. Fälle, bei denen mehr als ein Clusterzentrum in Frage kommt, werden auf beliebige, aber wiederholbare Weise entschieden. Die Indikatorfunktionen $\delta_i(\cdot)$ werden entsprechend angepasst.

2. **Neuberechnung der Clusterzentren** Setzen jedes Clusterzentrums $c_i \in C$ auf den Schwerpunkt der ihm momentan zugeordneten Datenpunkte $x \in \bar{X}_i$:

$$c_i \leftarrow \mu_i = \frac{\sum_j x_j \delta_i(j)}{\sum_j \delta_i(j)} \quad .$$

Mit jedem dieser Schritte werden die Kosten D_C monoton verringert. Der Algorithmus wird beendet, wenn sich keines der Clusterzentren mehr ändert oder wenn die Länge der größten Änderung eine vorgegebene Schwelle unterschreitet.

Der Algorithmus erfreut sich als einer der einfachen Basisalgorithmen der Clusteranalyse seit jeher einer großen Beliebtheit, obwohl er lediglich ein lokales Minimum von D_C findet, das Ergebnis stark von der initialen Position der Clusterzentren abhängt und lange Zeit keine theoretisch gesicherten Konvergenzaussagen verfügbar waren [114].

Der k-means Algorithmus kann als batch Version der Vektorquantisierung interpretiert werden. Bei der Vektorquantisierung wird ausgehend von initialen Startpositionen

der k Prototypen in jedem Schritt zufällig ein Datenpunkt $x \in \mathbb{R}^n$ gezogen. Dann wird der nächstliegende Prototyp p bestimmt – so dass $\forall p' \ ||p - x|| \leq ||p' - x||$ – und in Richtung des Datenpunktes bewegt:

$$p \leftarrow p + \alpha(x - p) = (1 - \alpha)p + \alpha x \ .$$

Wird die VQ auf einen festen, entsprechend der Wahrscheinlichkeitsdichtefunktionen zufällig gezogenen Datensatz von Datenpunkten $x \in X$ angewendet und werden die Aktualisierungen für alle Datenpunkte gleichzeitig berechnet und dann synchron, als Mittel aller Aktualisierungen durchgeführt (batch Aktualisierung), so ergibt sich für $\alpha = 1$ der k-means Algorithmus.

6.3.2 "Training" des irregulären Gitterapproximators

In Schritt drei des FQI-Algorithmus werden die Trainingsbeispiele $\mathcal{P} = \{s_t, a_t; \bar{q}_t | \ t = 1, \ldots, p\}$ verwendet, um den Funktionsapproximator mittels überwachten Lernens zu trainieren. Im Falle der Gitterapproximatoren besteht dieser Schritt darin, dass die Q-Werte an den Stützstellen auf den Mittelwert der entsprechenden Trainingsbeispiele im Einzugsbereich der jeweiligen Stützstelle gesetzt werden. Soll der Vektor q an dem Prototypen c_i aktualisiert werden, so werden für alle Aktionen $a_i \in A$ diejenigen Trainingsbeispiele aus $(s_t, a_t; \bar{q}_t) \in \mathcal{P}$ verwendet, bei denen die Aktion $a_t = a_i$ ausgewählt wurde. Sei die Menge dieser Trainingsbeispiele mit $\mathcal{P}_a = \{(s_t, a_t; \bar{q}_t) \in \mathcal{P} | \ a_t = a\}$ bezeichnet, dann wird der zur Aktion a_i gehörende Q-Wert q_i an der Stützstelle c_i auf den Mittelwert

$$q_i = \sum_{(s_t, a_t; \bar{q}_t) \in \mathcal{P}_a} \bar{q}_t \delta_i(t) \Big/ \sum_{(s_t, a_t; \bar{q}_t) \in \mathcal{P}_a} \delta_i(t) \tag{6.1}$$

gesetzt, wobei der Wert q_i unverändert bleibt, wenn es keine entsprechenden Trainingsbeispiele im Einzugsgebiet gibt, also $\sum_{(s_t, a_t; \bar{q}_t) \in \mathcal{P}_a} \delta_i(t) = 0$.

Das "Training" des Gitterapproximators geschieht also in einer einzigen "Epoche" und ist damit viel schneller als zum Beispiel das Training eines neuronalen Netzes durchzuführen. Wie schon in den Kapiteln 2 und 4 diskutiert, wird die Q-Funktion hier auf separaten Datenmengen approximiert, die jeweils nur die Trainingsbeispiele für eine bestimmte Aktion enthalten, und anschließend in unabhängigen Einträgen in den Stützstellen gespeichert.

6.3.3 Abfrage des irregulären Gitterapproximators

Der so konstruierte und trainierte Gitterapproximator kann auf zwei Weisen verwendet werden. Entweder kann bei Anfragen nach dem Q-Wert $\hat{Q}(s, a)$ an einer Stelle $s \in S$ zwischen den der Aktion a entsprechenden Werten q_i der umliegenden Stützstellen baryzentrisch interpoliert werden oder es kann der an der nahe gelegensten Stützstelle gespeicherte Wert zurückgegeben werden.

Interpolierende Gitteraproximatoren werden im Folgenden wie von Merke [97] als "lineare" Gitterapproximatoren bezeichnet. Mittels der Delaunay-Triangulation kann eine

durch die Prototypen definierte Partitionierung des Zustandsraumes in eine aus Dreiecken (bzw. Simplizes im Mehrdimensionalen) bestehende Partitionierung berechnet werden. Mit Hilfe der baryzentrischen Koordinaten des Zustandes s im ihn umgebenden Simplex kann dann ein gewichtetes Mittel über die an den Simplex aufspannenden Prototypen gespeicherten Q-Werte berechnet werden.

Die zweite Alternative ist die Verwendung "konstanter" Gitterapproximatoren [97]. In diesem Fall ist die Approximation $\hat{Q}(s,a)$ gleich dem zu a korrespondierenden Q-Wert an der an s nahe gelegendsten Stützstelle des Gitters. Dieses Vorgehen führt dazu, dass alle Zustände, die im Einzugsbereich vom selben Prototypen liegen, die gleichen "konstanten" Q-Werte für die Aktionen $a \in A$ besitzen. Der Einzugsbereich eines Prototpyen ist durch die Zellen in einem Voronoi-Diagramm definiert.

In den im Rahmen dieser Arbeit durchgeführten Versuchen werden aus folgenden Gründen ausschließlich die konstanten Gitterapproximatoren verwendet: Zunächst ist eine lineare Interpolation bei der Repräsentation nicht glatter Wertfunktionen – wie sie z.B. in kürzester Pfad Problemen auftreten – nicht hilfreich. Außerdem ist das Ziel, in jedem Fall durch den Approximator herrührende Stabilitätsprobleme auszuschließen, wie sie im Falle linearer Gitter von Ernst bei bestimmten Gittergrößen in der Praxis beobachtet [39] und von Merke für spezielle Aktualisierungen theoretisch begründet wurden [97].

6.3.4 Implementierung

Bei der Implementierung wurden verschiedene Methoden verwendet, um die Effizienz zu steigern. Zunächst werden kd-Bäume [75] eingesetzt, um den Zugriff auf die Zellen des Gitterapproximators und auch den k-means Algorithmus zu beschleunigen. Nach jeder Verschiebung der Prototypen wird ein neuer, balancierter kd-Baum für den schnellen Zugriff auf die M Prototypen konstruiert. Dazu wird eine zu [119] ähnliche Implementierung mit einer Aufteilung der Hyperzellen durch den Median der enthaltenen Prototypen verwendet. Der mit der Konstruktion verbundene Rechenaufwand ist von der Komplexität $O(M\,log\,M)$ [119]. Alle Zugriffe auf Zellen des konstanten Gitters, die dem Auffinden des nächst gelegenen Prototypen entsprechen, sind danach von der Komplexität $O(log\,M)$ und auch Bereichsabfragen für andere Clusteringverfahren sind sehr effizient möglich [75].

Für jeden vollständigen DP-Approximationsschritt (Schritt 2 und 3 in FQI) müssen im Falle von N Transitionen $2N$ Zugriffe auf das Gitter vorgenommen werden; N Zugriffe, um die Kosten $\hat{V}^i(s')$ an den Endzuständen s' der Transitionen (s,a,r,s') herauszusuchen, und N Zugriffe, um die berechneten Q-Werte $\bar{q}^{i+1}$ an den zu den Startzuständen s nahe gelegensten Prototypen aufzusummieren. Anschließend müssen an allen M Prototypen Mittelwerte über die aufsummierten Q-Werte gebildet werden. Da bis zur Konvergenz des FQI-Algorithmus einige Schleifendurchläufe nötig sind, kann dieser Vorgang extrem beschleunigt werden, indem vor Beginn des dynamischen Programmierens eine bidirektionale Verkettung aller Zustände in $\mathcal{F}$ mit den zugehörigen Prototypen erzeugt wird. Es gibt an jedem Prototyp nach Aktionen geordnete Listen mit Zeigern auf alle zugehörigen Transitionen. Gleichzeitig werden an jeder Transition Zeiger auf

die zu dem Start- und zu dem Endzustand zugehörigen Prototypen gespeichert. Die einmalige Konstruktion der Verzeigerung hat eine Komplexität von $O(N \log M)$, alle folgenden Zugriffe auf die nahe gelegensten Prototypen sind ab dann nur noch von der Komplexität $O(1)$.

Zusätzlich wurden sowohl bei der Implementation des k-means Verfahrens als auch bei der Implementation des dynamischen Programmierens und des Trainierens der Gitter die gleichen Methoden zur Parallelisierung wie bereits zuvor bei den tiefen neuronalen Netzen (siehe Abschnitt 5.2.2) eingesetzt: Summen und insbesondere die Abstände werden mittels BLAS berechnet, die Summation der Positionen der Datenpunkte im k-means-Verfahren und der Q-Werte beim Training des Approximators werden auf n Kopien der Prototypen parallel berechnet. Hierzu werden die Daten wieder in n Teilmengen eingeteilt, die von je einem von n Threads bearbeitet werden. Die erzeugten Teilsummen werden wie bei den neuronalen Netzen anschließend aufsummiert.

6.3.5 Konvergenzverhalten

Die hier beschriebenen irregulären Gitterapproximatoren sind Averager im Sinne der Definition 1 und lassen sich auch in der kernelbasierten Form nach Definition 5 (beide Kapitel in 4) beschreiben. Daher konvergieren sie entsprechend des Ergebnisses für KADP für $\gamma < 1$ zu einem eindeutigen Fixpunkt.

Um die Formulierung mittels der Kernel deutlich zu machen, wurde in der Beschreibung des k-means Algorithmus durchgängig die Indikatorfunktion $\delta_i(\cdot)$ verwendet. Die Gewichte, mit der die Trainingsbeispiele $(s_t, a_t; \bar{q}_t) \in \mathcal{F}_a$ bei der Mittelwertbildung (6.1) eingehen, genügen offensichtlich den Anforderungen (4.34) und (4.35). Für die hier betrachteten konstanten Gitter erhalten alle Zustände in dem Einzugsbereich des Zentroiden c_i, definiert durch die ihn umgebende Zelle im Voronoi-Diagramm, den gleichen approximativen Wert zugewiesen – innerhalb des Einzugsbereichs werden also für alle approximierten Q-Werte $\hat{Q}(s, a)$ die gleichen Gewichte über die Trainingsbeispiele verwendet.

6.4 Empirische Evaluation

Für die Evaluation des ClusterRL-Algorithmus wird das Mountain-Car verwendet. Das Moutain-Car-Problem mit seinen Diskontinuitäten in der Wertfunktion ist gut geeignet, die Güte der Strategie und insbesondere der approximierten Wertfunktion in einer von den Anforderungen zum Einsatz in DFQ ähnlichen Situation zu beurteilen.

In dieser Evaluation werden drei Experimente mit den irregulären Gittern durchgeführt. Zunächst wird versucht, die optimale Wertfunktion auf Basis von unterschiedlich großen Mengen von zufällig gezogenen Traningsbeispielen in Gitterapproximatoren abzuspeichern, wie es im Schritt 3 von FQI im idealen Fall der Konvergenz zur optimalen Wertfunktion in der letzten Iteration passieren würde. Dieses Experiment dient dazu, die Approximationseigenschaften der irregulären Gitter mit den regulären Gittern zu vergleichen und dabei alle Einflüsse auszuschließen, die durch das dynamische Program-

mieren selbst entstehen. Anschließend wird ein Experiment durch Anwendung des vollständigen FQI-Algorithmus auf verschieden große Datensätze durchgeführt, wobei die Varianten zur Initialisierung und Ausdünnung der Gitterstruktur untersucht werden. Im abschließenden Experiment wird eine Version des Mountain-Car-Problems verwendet, bei dem die Zustände auf die Mantelfläche eines Zylinders in $\mathbb{R}^3$ projiziert werden. In dieser Situation, die der Anordnung der Daten in dem Carrerabahn-Experiment (siehe Abschnitt 5.3.7) ähnelt, sollten sich die Vorteile der irregulären Gitter in sehr ungleichmäßig ausgenutzten Zustandsräumen zeigen.

6.4.1 Mountain-Car

Bei dem Mountain-Car-Problem [101] befindet sich ein Wagen in einem tiefen Tal und muss ein Ziel auf einem Hügel erreichen (siehe Abbildung 6.4). Weil die Antriebskraft des Wagens nicht ausreicht, die Steigung zum Ziel direkt zu überwinden, muss er zunächst vom Ziel weg beschleunigen und sich so unter Ausnutzung der Gravitation "aufschwingen". Das hier verwendete Mountain-Car orientiert sich an der in [102] beschriebenen Variante ("Puck on the hill"), die deutlich von der von Sutton verwendeten Vereinfachung [155] abweicht.

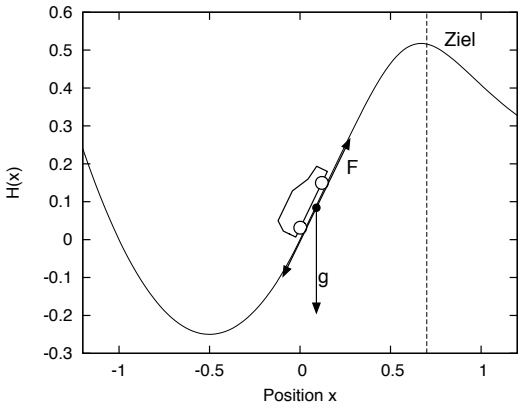


Abbildung 6.4: Das Mountain-Car-Problem.

Die Oberfläche der Fahrbahn ist wie folgt durch die Funktion H gegeben:

$$G(x) := \begin{cases} x^2 + x & : \quad x < 0 \\ \frac{x}{\sqrt{1+5x^2}} & : \quad x \geq 0 \ . \end{cases}$$

Sei x die Position des Wagens und α der Winkel zwischen der Oberfläche der Fahrbahn und einer horizontalen Linie, dann ist die Dynamik des Systems durch folgende Differentialgleichung gegeben:

$$\ddot{x} = -g \sin(\alpha) \cos(\alpha) + \frac{F}{m} \cos(\alpha) - \frac{\dot{x}^2}{\cos(\alpha)^2} \frac{G'(x)G''(x)}{(1 + (G'(x))^2)^2} \ ,$$

wobei m die Masse des Wagen, F die vom Antrieb des Wagens ausgeübte Kraft und g die Gravitationskonstante bezeichne. Diese in CLSquare implementierte Dynamik ist unter physikalischen Gesichtspunkten noch realistischer als die in [102] angegebene und berücksichtigt auch die zweite Ableitung von G.

Alle Versuche wurden mit $m = 1\,\text{kg}$ und $g = 9.81\,\text{m/s}^2$ durchgeführt. Der Agent erhält den zweidimensionalen Zustandsvektor $s_t = (x_t, \dot{x}_t)$ und entscheidet sich zu den diskreten Zeitschritten $t = 0, 1, ..., n$ mit fester Zeitdifferenz $\Delta t = 0.05\,\text{s}$ für eine von drei Aktionen $a \in \{-4, 0, +4\}\,\text{N}$.[1] Die Position x und Geschwindigkeit $\dot{x}$ zum Zeitpunkt $t + 1$ wird mittels des vierstufigen Runge-Kutta-Verfahrens [119] bestimmt. Der absorbierende Zielbereich ist erreicht, wenn die Position $x \geq 0.7$. In diesem Fall wird die Episode beendet. Verlässt der Wagen den zulässigen Wertebereich von $x \in [-1.2, +1.2]$ oder erreicht er das Ziel nicht innerhalb von 200 Zeitschritten, wird die Episode ebenfalls abgebrochen. Es wird folgende deterministische Belohnungsstruktur eingesetzt, die ein kürzester Pfad Problem definiert:

$$r_{t+1} = R(s_t, a_t, s_{t+1}) := \begin{cases} 0 & x_t \geq +0.7 \\ -1000 & x_t < -1.2 \\ -1 & sonst \end{cases}$$

Die einzige Methode, unter dieser Belohnungsstruktur Kosten zu vermeiden, ist es, möglichst schnell zum absorbierenden Zielzustand zu gelangen, nach dessen Erreichen keine weiteren Kosten anfallen.

Das Mountain-Car ist ein niedrigdimensionales, kontinuierliches Problem. An dieser Stelle ist es aufgrund folgender Eigenschaften besonders gut als Benchmark geeignet:

- Es gibt einen Flaschenhals auf dem Weg zum Ziel, bei dessen Durchschreiten die ein- oder zweimalige Auswahl einer nicht optimalen Aktion zum "Zurückfallen" in den Startbereich führt.

- Es gibt daher im Zustandsraum scharfe Diskontinuitäten, an denen die Anpassungsfähigkeit der irregulären Gitter getestet werden kann.

- Aufgrund der Geschwindigkeit sind die "Schrittlängen" des Agenten im Zustandsraum sehr unterschiedlich. Wegen der relativ niedrigen Startgeschwindigkeit liegen die Observationen im Startbereich dichter beisammen als in den "äußeren" Bereichen der Wertfunktion, wo der Agent entweder aufgrund der hohen Geschwindigkeit in großen Schritten seine Position ändert oder durch die Gravitation stark beschleunigt wird (gut zu erkennen in Abbildung 6.5, rechts).

- Die Gravitation zieht den Agenten immer wieder auf Spiralbahnen zur Ausgangsposition in der Talsohle zurück, so dass – anders als in der Grid-World – eine Zufallsstrategie keine gleichmäßige Abdeckung erzielt.

- Trajektorien, Wertfunktion, Strategie und die Lage der Daten und Stützstellen sind gut im zweidimensionalen Zustandsraum zu visualisieren.

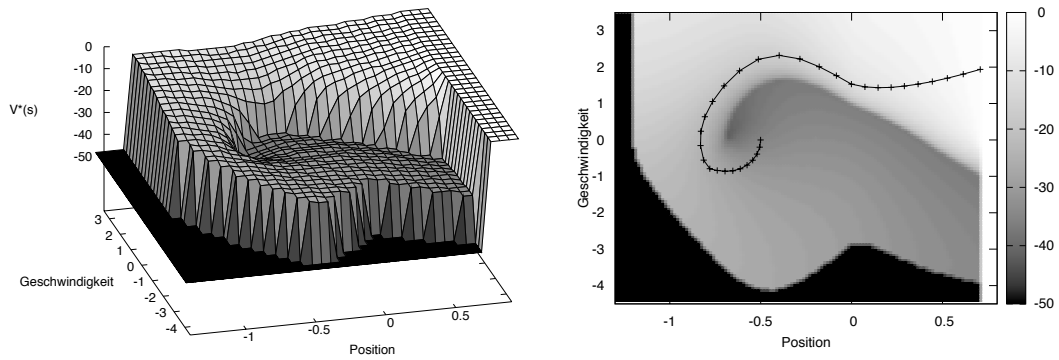


Abbildung 6.5: Optimale Wertfunktion $V^*(s)$ des Mountain-Cars und optimale Trajektorie vom Tal ins Ziel (durch schwarze Linie verbundene Zustände im rechten Bild), bestimmt auf 6.25 Mio Stützstellen mittels approximativer Wertiteration nach Gordon [51]. Zu Visualisierungszwecken wurde die Wertfunktion weiter diskretisiert (links: 1600 Vertices, rechts: 1 Mio Vertices) und in beiden Bildern der Wertebereich bei -50 abgeschnitten.

Die in Abbildung 6.5 dargestellte optimale Wertfunktion V^* des hier beschriebenen Problems ist nicht stetig, sondern weist wegen der diskreten Zeitschritte Stufen auf. Die Stufen entsprechen dabei der Anzahl der Schritte, die unter optimaler Strategie noch zum Ziel benötigt werden. Von allen Startzuständen, die nicht unweigerlich zu einem Verlassen des Arbeitsbereichs führen, ist der Zielbereich in maximal 39 Zeitschritten erreichbar, von der Talsole bei $x = -0.5$ werden aus dem Stand 32 Schritte benötigt.[2] Schwarz markiert in den Diagrammen den Zustandsbereich, der entweder außerhalb des zulässigen Wertebereiches für die Position x liegt oder aus dem es aufgrund der hohen negativen Geschwindigkeit $\dot{x}$ kein Entkommen mehr gibt.

6.4.2 Approximation der optimalen Zustands-Wertfunktion $V^*(s)$

Zunächst sollen die irregulären mit den regulären Gittern in einem grundsätzlichen Experiment verglichen werden. Um die Güte der Approximation der optimalen Wert-

[1] In der Folge wird auf die Angabe von SI-Basiseinheiten und abgeleiteter Einheiten verzichtet.

[2] Abseits vom eigentlichen Untersuchungsziel ist eine interessante Beobachtung, dass selbst bei Verwendung von über 6 Millionen Stützstellen zur Approximation der optimalen Wertfunktion – üblich sind in der Literatur eher um 1000 bis 10000 Stützstellen [53, 166] – und eines modellbasierten Verfahrens noch signifikante Fehler auftreten. Zwar führt die gierige Auswertung der hier approximierten optimalen Wertfunktion von allen getesteten zufälligen Startzuständen zu einem optimalen Verhalten. Jedoch weist die Schätzung der Kosten selbst noch Fehler auf: Sogar entlang der optimalen Trajektorie vom Tal des Labyrinths $s = (-0.5, 0)$ bis ins Ziel kommt es zu einem Sprung in den geschätzten Pfadkosten: Von Schritt 20 auf 21 ändern sich die erwarteten Kosten nicht, um dann von Schritt 21 auf 22 gleich um 2 Schritte zu fallen. Dies deutet darauf hin, dass die Seitenmaße der vewendeten rechtwinkligen Hyperzellen selbst bei 6.25 Millionen Stützstellen immer noch zu lang sind, und es an manchen ungünstigen Stellen im Zustandsraum möglich ist, dass der Agent unter Anwendung der optimalen Aktion wieder in derselben Zelle landet.

funktion $V^*(s)$ selbst ohne die Einflüsse des dynamischen Programmierens untersuchen zu können, wird eine feste Menge von Trainingsbeispielen $(s; V^*(s))$ der in Abbildung 6.5 dargestellten, auf 6.25 Mio Stützstellen mittels FVI angenäherten optimalen Wertfunktion verwendet. Die zu testenden Approximatoren werden dann auf einem Teil dieser Beispiele trainiert und auf den anderen hinsichtlich der Generalisierungsleistung getestet. Die Güte der quasi unter optimalen Bedingungen, nämlich unter Kenntnis der tatsächlichen, optimalen Wertfunktion $V^*(s)$ erzielten Approximation gibt einen Hinweis darauf, was später in der Kombination des Approximators mit Techniken des dynamischen Programmierens maximal erreicht werden kann. Zwar könnten die Eigenschaften der Approximatoren prinzipiell auch an jeder beliebigen Funktion verglichen werden, die Approximation einer wirklichen Wertfunktion erhöht aber die Aussagekraft für die spätere Anwendung innerhalb von FQI. Da die Strategie von der Wertfunktion abgeleitet wird, wirkt sich die Approximationsgüte direkt auf die Qualität der Strategie aus und wird in diesem Experiment auch durch gierige Auswertung des Resultats der Approximation getestet.

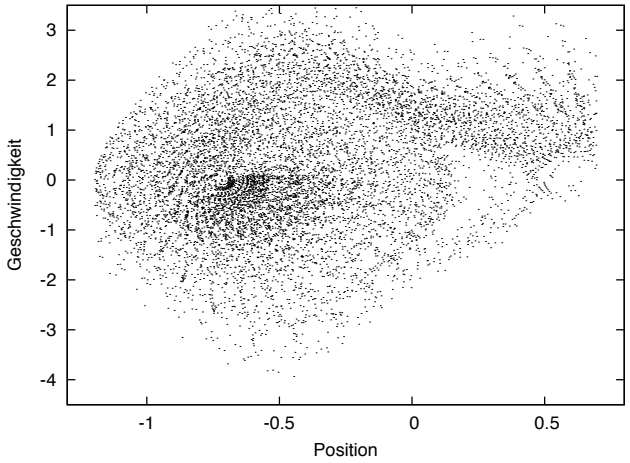


Abbildung 6.6: Verteilung der Zustände in der Stichprobe mit $p = 10\,000$ Übergängen bei Anwendung der ϵ-greedy Strategie mit kleiner werdendem ϵ.

Vorgehensweise Die Trainingsbeispiele werden so in der Interaktion mit dem System gesammelt, dass ihre Verteilung der zu erwartenden Verteilung in einem richtigen Lernvorgang ähnelt. Die Startzustände liegen alle im Bereich $(x, \dot{x}) \in [-1, 0] \times [-1, +1]$. Aktionen werden mittels ϵ-greedy von der bereitgestellten, optimalen Wertfunktion abgeleitet. Zunächst wird dabei mit einer reinen Zufallsstrategie agiert ($\epsilon = 1$), die dann nach und nach durch lineare Absenkung des Wertes von ϵ in die optimale Strategie übergeht. Jeder in dieser Interaktion beobachtete Zustand $s = (x, \dot{x})$ wird zusammen mit dem optimalen Zustandswert $V^*(s)$ als ein Trainingsbeispiel $(s; V^*(s))$ abgespeichert. Die auf diese Weise erzeugte Menge $\mathcal{P} = \{(s_t; V^*(s_t)) | t = 1, \ldots, p\}$ von p Trainingsbeispielen spiegelt die Verteilung wider wie sie beim online-Lernen auftreten würde, wo

der Agent zunächst ohne Wissen zufällig und dann mit zunehmendem Lernfortschritt immer zielgerichteter interagiert. Die Verteilung im Zustandsraum einer Stichprobe mit $p = 10\,000$ Trainingsbeispielen ist in Abbildung 6.6 zu sehen.

Auf solchermaßen erzeugten, verschieden großen Stichproben mit 1000 bis 100 000 Trainingsbeispielen werden sowohl reguläre als auch irreguläre Gitterapproximatoren unterschiedlicher Auflösung (Anzahl der Zellen) trainiert. Dabei werden bei den irregulären Gittern noch keine der beiden Varianten zur Ausdünnung des Gitters und zur Initialisierung zusätzlicher Prototypen an den Terminalzuständen verwendet. Präsentiert werden jeweils die Ergebnisse einer 10-fach Kreuzvalidierung.

Auswertungskriterien Folgende, für den späteren Einsatz in DFQ relevante Kriterien dienen zur Bewertung der approximierten Funktionen:

Regressionsfehler Auf der nicht für das Training verwendeten Testmenge wird zunächst – wie bei der Beurteilung von Regressionsergebnissen üblich – der durchschnittliche quadratische Fehler (MSE) zwischen Approximation $\hat{V}(s)$ und Zielfunktion $V^*(s)$ bestimmt:

$$\text{MSE} = \sum_{(s;V^*(s))\in\mathcal{P}} (\hat{V}(s) - V^*(s))^2 \Big/ |\mathcal{P}| \ .$$

Aufgrund der Berechnung an den Elementen der Stichprobe, die in der Interaktion mit dem System beobachtet wurden, bezieht sich dieser Fehler auf Stellen des Zustandsraums, die wirklich erreicht werden können die und für die Strategie besonders relevant sind.

Klassifikationsfehler Der Anteil der Zustände in der Testmenge s, bei denen der Approximationsfehler $|\hat{V}(s) - V^*(s)|$ die Kosten eines halben Schrittes im kürzester Pfad Problem überschreitet, ist ein nützliches Bewertungskriterium der Genauigkeit und Generalisierungsleistung des Approximators mit einem engeren Bezug zum Reinforcement Lernen und zur Qualität einer abgeleiteten Strategie.

Akkumulierte Pfadkosten Von der approximierten Zustandswertfunktion wird über eine gierige Auswertung eine Strategie abgeleitet und auf einer festen Menge von 1 000 zufällig ausgewählten Startzuständen evaluiert. Besonderer Wert wird darauf gelegt, dass sich bereits diese Startzustände von den beim Ziehen der Stichprobe verwendeten Startzuständen unterscheiden, um von vornherein jegliche positivverfälschende Einflüsse auszuschließen, die durch das Abschreiten identischer Trajektorien im Training und beim Testen entstehen könnten. Ein Vergleich der akkumulierten Kosten erlaubt eine erste Beurteilung der unterschiedlichen Verfahren hinsichtlich der zu erwartenden Strategiegüte.

Anteil zielführender Trajektorien Neben den akkumulierten Pfadkosten wird auch getestet, wie hoch der Anteil der zielführenden Trajektoren ist.

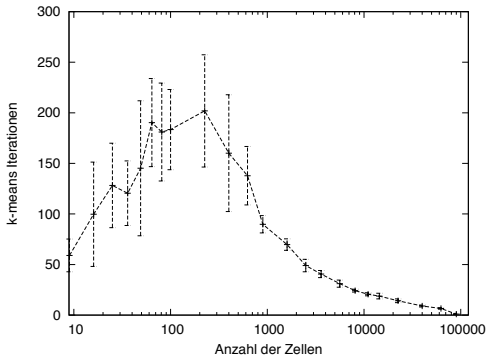


Abbildung 6.7: Anzahl der Iterationen, die der k-means Algorithmus im Mittel zur Konvergenz in Abhängigkeit von der Anzahl der Zentren auf 100 000 Datenpunkten benötigt. Angaben mit Standardabweichung in der Kreuzvalidierung. Logarithmische Skala an der x-Achse.

Ergebnisse In Abbildung 6.7 ist die Anzahl der Iterationen zu sehen, die Lloyds Algorithmus mit verschieden vielen Prototypen auf 100 000 Datenpunkten bis zur Konvergenz benötigt. Der interessante Bereich liegt zwischen 2 bis maximal 100 Datenpunkten je Gitterzelle (entspricht hier 1000 bis 50 000 Gitterzellen). Hier konvergiert der Algorithmus überlicherweise in unter 100 Schritten, und auch die Schwankung zwischen den einzelnen Wiederholungen ist in diesem Bereich sehr niedrig. Um aber extrem lange Rechenzeiten auch in ungünstigen Situationen sicher auszuschließen, wird Lloyds Algorithmus in allen folgenden Experimenten nach spätestens 500 Schritten abgebrochen, was der doppelten Menge Interationen entspricht, die im ungünstigsten Fall bei dieser Stichprobengröße benötigt werden.

Die von den irregulären und regulären Gittern unterschiedlicher Größe erzielten Regressionsfehler auf der Stichprobe mit 100 000 Beispielen sind in Abbildung 6.8 dargestellt. Es sind drei Dinge zu erkennen:

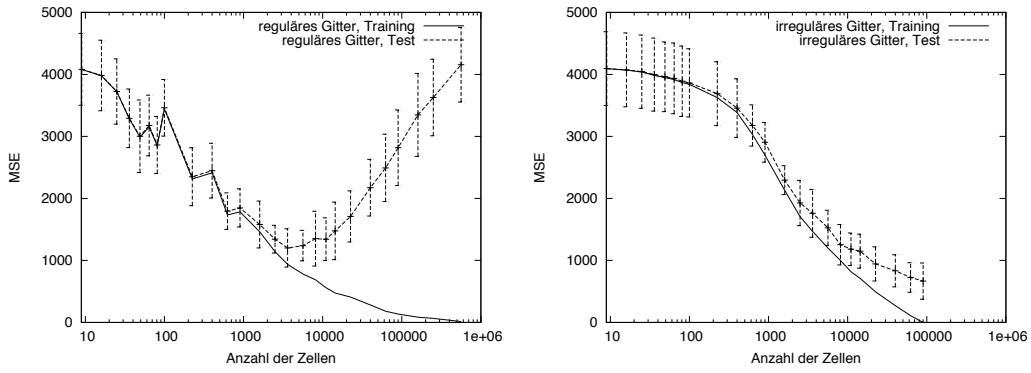


Abbildung 6.8: Durchschnittlicher Fehler auf Trainings- und Testdaten bei Verwendung unterschiedlich feiner Gitter mit Darstellung der Standardabweichung des Testfehlers in der Kreuzvalidierung auf einer Stichprobe mit 100 000 Datenpunkten. Links: Reguläre Gitter. Rechts: Irreguläre Gitter. Logarithmische Skala an der x-Achse.

1. Die irregulären Gitter erzielen bei optimaler Gittergröße auf diesem Datensatz einen deutlich niedrigeren MSE in der Kreuzvalidierung.

2. Während der durchschnittliche Validierungsfehler bei den irregulären Gitterapproximatoren kontinuierlich mit der Erhöhung der Zellenzahl absinkt, gibt es bei den regulären Gittern bis zu einer Größe von 1000 Zellen Auflösungen, die zu besonders schlechten Ergebnissen führen. So ist das Ergebnis eines Gitters mit 12×12 Zellen deutlich schlechter als bei Verwendung von nur 10×10 Zellen.

3. Bei den regulären Gittern steigt der Regressionsfehler anders als bei den irregulären Gittern ab einer gewissen Größe wieder an. Hierzu ist anzumerken, dass bei den regulären Gittern auch Zellenzahlen getestet werden müssen, die höher als die Anzahl der Datenpunkte sind – man weiß nämlich nicht, wo genau und wie Dicht die Daten liegen. Bei den irregulären Gittern wird die Struktur dagegen durch die Daten definiert, so dass die Anzahl der Datenpunkte eine ganz natürliche Obergrenze für die Anzahl der Gitterzellen ist.[1]

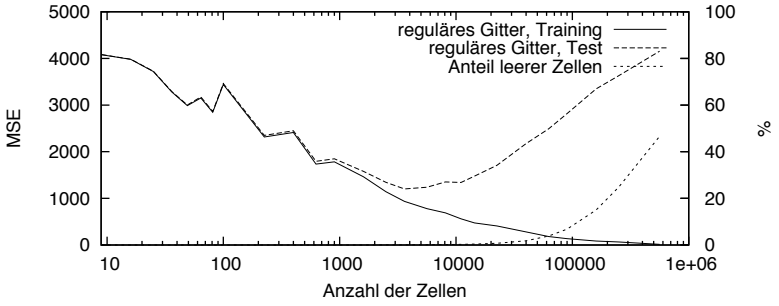


Abbildung 6.9: Durchschnittlicher Regressionsfehler bei Verwendung des regulären Gitters (aus Abbildung 6.8) und prozentualer Anteil der Testpattern, die in eine Zelle fallen, die von keinem der Trainingspattern getroffen wird. Logarithmische Skala an x-Achse.

Der Anstieg des Fehlers bei den regulären Gittern ab einer Gittergröße von über 3000 Zellen lässt sich dadurch erklären, dass mit zunehmender Auflösung die Wahrscheinlichkeit zunimmt, dass in einzelnen Zellen kein einziger Datenpunkt der Trainingsmenge liegt, ihnen deshalb kein Wert zugewiesen wird, sie aber von einem Datenpunkt der Testmenge getroffen werden. Dieser Effekt ist gut in Abbildung 6.9 zu erkennen, wo der Anteil der Testpunkte, die in eine "leere" Zelle fallen, gegen die Trainings- und Testfehler aufgetragen ist. Zunächst steigt die Genauigkeit in der Regression mit der Auflösung. Ab einer gewissen Auflösung – die in diesem Fall deutlich unter der Menge der Trainingsdaten liegt – erhöht sich dann aber der Anteil der im Training nicht aktualisierten Zellen, was sich schließlich zunehmend auf den Validierungsfehler auswirkt. Die irregulären Gitter dagegen sind nicht auf eine regelmäßige Zellstruktur angewiesen und daher

[1] In den Diagrammen werden zu Vergleichzwecken dennoch immer identische x-Achsen verwendet, auch wenn dadurch der Graph der irregulären Gitter bedingt durch die Menge der Trainingsdaten wie in Abbildung 6.8 "vorzeitig" endet.

nicht von diesem Problem betroffen. Es werden maximal so viele Prototypen verwendet, wie Trainingsdaten vorhanden sind – in diesem Fall ist das Gitter identisch zu einem nächster-Nachbar-Klassifikator. Daher wird jede Zelle mindestens einmal "getroffen".

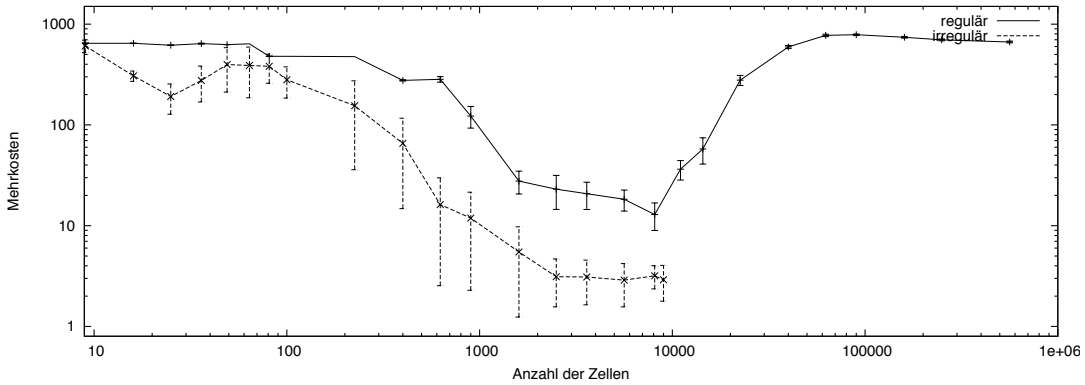


Abbildung 6.10: Gegenüber der optimalen Strategie entstehende Mehrkosten bei Ableitung gieriger Strategien von unterschiedlich großen regulären und irregulären Gittern. Ergebnisse einer Kreuzvalidierung mit 10 000 Datenpunkten. Logarithmische Skalen an beiden Achsen.

Besonders deutlich wird der Unterschied zwischen regulären und irregulären Gittern auf kleineren Datensätzen. In Abbildung 6.10 sind die durchschnittlichen Kosten zu sehen, die mit einer von den trainierten Gittern abgeleiteten gierigen Strategie erzielt werden. Die in der Kreuzvalidierung verwendete Stichprobe enthält hier nur 10 000 Datenpunkte. Dargestellt sind wie in allen weiteren Graphen der Kosten die im Durchschnitt über alle Startzustände entstehenden Mehrkosten gegenüber der Verwendung der optimalen Strategie. Das Ergebnis der irregulären Gitter bei optimaler Gittergröße ist um fast eine Größenordnung besser (ca. 2 zusätzliche Schritte gegenüber den mehr als 10 zusätzlichen Schritten bei Verwendung der regulären Gitter). Allerdings weisen die von den irregulären Gittern abgeleiteten Strategien eine vergleichsweise hohe Schwankung auf, insbesondere im Bereich von 200-2000 Zellen.

Die Ergebnisse einer ausführlichen Versuchsreihe von Kreuzvalidierungen über unterschiedlich große Stichproben und Gittergrößen von 9 (regulär: 3×3) bis 1 000 000 (regulär: 1000×1000) Zellen sind in Abbildung 6.11 dargestellt. Die irregulären Gitter sind nach allen drei Kriterien – akkumulierte Pfadkosten, Anteil zielführender Strategien und Klassifikationsfehler – den regulären Gittern überlegen. Der Bereich um die optimale Gittergröße, in der gute Ergebnisse erzielt werden, ist bei den irregulären Gittern breiter als bei den regulären Gittern. Zudem verschlechtern sich die Ergebnisse bei zu groß gewählten Gittern nicht wesentlich.

In Tabelle 6.1 sind die Ergebnisse dieser Versuchsreihe noch einmal zusammgefasst, wobei nur die jeweils beste Gittergröße berücksichtigt wird. Die Unterschiede sind umso deutlicher, je kleiner der Datensatz ist. Auf 1000 Datenpunkten erreichen die von den irregulären Gittern abgeleiteten Strategien bereits von über 98% der getesteten Startpositionen das Ziel, während die Strategien von den regulären Gittern nur in 64% der

Fälle ins Ziel führen und auch noch oft den zulässigen Wertebereich verlassen. Selbst bei Verwendung von 100 000 Datenpunkten sind die irregulären Gitter noch knapp besser; erst bei 1 Mio Datenpunkten verschwinden die Unterschiede.

Tabelle 6.1: Zusammenfassung der Experimente im Mountain-Car: Übersicht über die mit optimaler Auflösung erreichten Klassifikationsfehler (CER), durchschnittlichen Pfadkosten (CST) und den Anteil der ins Ziel führenden Testtrajektorien (RG) für reguläre (R) und irreguläre (I) Gitter auf verschieden großen Datensätzen (Anzahl der enthaltenen Übergänge bzw. Trajektorien in den ersten beiden Spalten).

Überg.	Traj.	CER I	CER R	CST I	CST R	RG I	RG R
1000	11	46.60%	51.20%	50.02	418.65	98.53%	63.59%
10000	142	19.17%	27.74%	30.19	40.20	99.88%	98.94%
100000	1320	8.54%	14.78%	29.12	29.18	99.99%	99.97%
1000000	13308	6.77%	7.82%	27.33	27.26	100.00%	100.00%

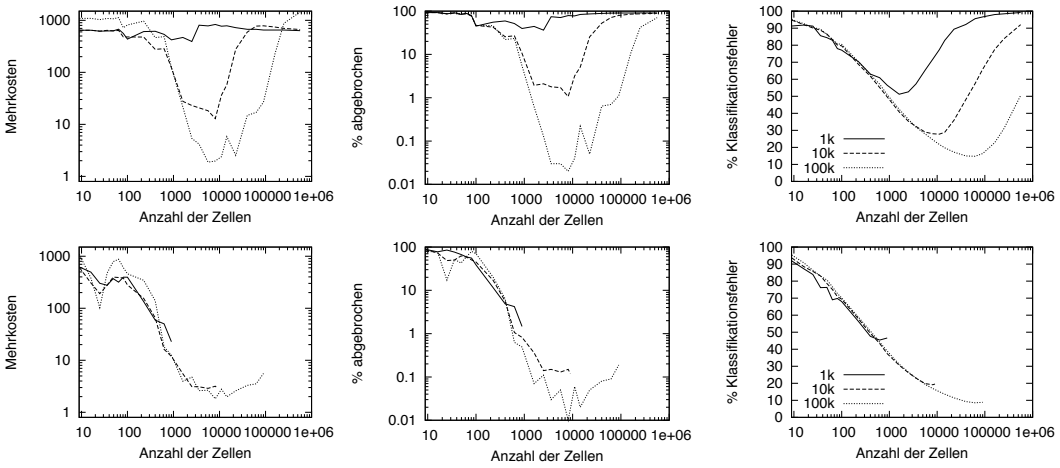


Abbildung 6.11: Kennlinien verschieden feiner regulärer (oben) und irregulärer Gitter (unten) auf drei unterschiedlich großen Stichproben. Links: Durchschnittliche Mehrkosten gegenüber der optimalen Strategie. Mitte: Anteil der nicht im Ziel endenden Trajektorien. Rechts: Klassifikationsfehler hinsichtlich der erwarteten Schritte bis zum Ziel. Teilweise logarithmische Achsen.

Diskussion Irreguläre Gitterapproximatoren sind den regulären Gitterapproximatoren insbesondere auf den kleinen Datensätzen überlegen. Die bei optimaler Gittergröße erzielten Ergebnisse sind durchweg besser. Ein Vorteil ist auch, dass die erzielten Ergebnisse etwas weniger abhängig von der gewählten Anzahl der Zellen sind. Außerdem muss anders als bei den regulären Gittern keine Abwägung zwischen Genauigkeit und leeren Gitterzellen getroffen werden. Die zu große Erhöhung der Auflösung stellt bei

den irregulären Gittern abgesehen vom zusätzlichen Speicherbedarf kein Problem dar. Nicht gänzlich unproblematisch ist allerdings die relativ hohe Varianz der erzielten Ergebnisse im Bereich der interessanten Gittergrößen. Die Schwankungen rühren von den unterschiedlichen Gitterstrukturen her, die durch die Anwendung des k-means Algorithmus je nach initialer Lage der Prototypen erzeugt werden. In der Praxis könnten diese Schwankungen eine mehrmalige Wiederholung des Lernvorganges von unterschiedlichen Initialisierungen erforderlich machen.

6.4.3 Erprobung der Trainingsvarianten

Als ein Problem der irregulären Gitter wurde im vorhergehenden Experiment die höhere Varianz der Ergebnisse und die damit verbundene Abhängigkeit der erzielten Strategiequalität vom Zufall identifiziert. Außerdem waren die Ergebnisse stark von der gewählten Größe des Gitters abhängig. Zwar ist der Bereich, in dem gute Ergebnisse erzielt werden, größer als bei den regulären Gittern, aber wenn zu wenige Prototypen gewählt werden, sind die Ergebnisse weit vom Optimum entfernt. Eine größere Unabhängigkeit von diesem Parameter wäre hier insbesondere im Hinblick auf den Einsatz in DFQ wünschenswert, wo dieser Parameter nicht für jeden Merkmalsraum aufwändig manuell bestimmt werden kann. Für beide Probleme bieten die bereits beschriebenen Trainingsvarianten mit einer automatischen Ausdünnung der Gitter und dem Hinzufügen zusätzlicher Prototypen auf den Übergängen in den Zielzustand praktikable Lösungen.

Vorgehensweise Zum Test der Trainingsvarianten wird der ClusterRL-Algorithmus auf einen relativ großen Datensatz von 30 000 Transitionen angewendet, wobei die Übergänge auf die gleiche Weise wie im vorhergehenden Experiment gesammelt werden. Kreuzvalidiert werden zunächst drei unterschiedliche Werte für den Parameter k_{init} – dieser Parameter gibt die anfänglichen Gittergrößen an, siehe Abschnitt 6.3 – mit und ohne Anwendung der Trainingsvarianten. Während die Anzahl der Gitterzellen bisher nicht automatisch angepasst wurde, wird sie beim Einsatz der Trainingsvarianten von dieser vorgegebenen Ausgangsgröße adaptiert. Der Parameter für die initiale Gittergröße wird dabei so gewählt, dass sowohl ein zu kleines Gitter, mit dem sich ohne die Trainingsvarianten keine guten Ergebnisse erzielen lassen, ein mittelgroßes Gitter mit guten Ergebnissen und ein zu großes Gitter mit wiederum schon schlechteren Ergebnisssen vertreten sind.

Ergebnisse In Tabelle 6.2 sind die Auswirkungen der Anwendung der Heuristiken zur Gitterausdünnung (“thin”) und zum Hinzufügen von Prototypen (“add”) auf die Güte der erlernten Strategie zu erkennen. Das Ausdünnen der Daten führt zu einer leichten Verbesserung der Kosten bei den mittelgroßen und zu einer deutlichen Verbesserung bei den großen Gittern, während bei dem zu kleinen Gitter – erwartungsgemäß – keine Verbesserung durch weiteres Entfernen von Prototypen erzielt werden kann. Ein weiterer Pluspunkt ist die deutliche Verringerung der Varianz der Ergebnisse bei den mittelgroßen und großen Gittern. Das Hinzufügen weiterer Prototypen an den Übergängen in

Terminalzustände führt dagegen vor allem zu einer Verbesserung bei den zu kleinen Gittern, wo die durchschnittlichen Kosten deutlich von 45.29 auf 37.88 abgesenkt werden können. Besonders herauszustellen ist hier die erzielte große Verringerung der Varianz der Ergebnisse auf den zu kleinen Gittern. Die Kombination der beiden Verfahren führt zu deutlichen Verbesserungen bei den Kosten in allen drei Gittergrößen. Die Unterschiede zwischen den initialen Gittergrößen nehmen erkennbar ab, die Schwankungen in den Ergebnissen der Kreuzvalidierung werden durchweg gegenüber dem Basisalgorithmus mit fester Gittergröße halbiert.

Tabelle 6.2: Vergleich des Effekts der Initialisierung von zusätzlichen Prototypen auf allen Übergängen in den Zielzustand (add) und der Ausdünnung (thin). Durchschnittliche Kosten mit Standardabweichung abgeleiteter Strategien für $k = 225, 900, 2500$.

	225	900	2500
OHNE	45.29 ± 18.2	34.04 ± 5.34	36.63 ± 5.93
ADD	37.88 ± 3.54	34.19 ± 5.47	35.19 ± 4.65
THIN	45.29 ± 18.2	33.12 ± 3.83	32.17 ± 2.96
BEIDE	39.59 ± 9.86	32.00 ± 2.13	32.53 ± 2.79

Auf einem Datensatz von 30 000 Transitionen, der bereits groß genug ist, um auch mit den regulären Gittern gute Ergebnisse zu erzielen, wird der durch die automatische Adaptation der Gittergröße hinzugewonnene Vorteil gegenüber den regulären Gittern mit fester Gittergröße deutlich. Getestet werden initiale Gittergrößen k von 9 bis 30 000 bzw. bis 1 Mio. bei den regulären Gittern. Während die durchschnittlichen Kosten bei optimaler Gittergröße nur weniger als zwei Zehntel auseinander liegen, ist in Abbildung 6.12 gut zu sehen, wie die finalen Gittergrößen unabhängig von der Wahl von k automatisch näher in Richtung der optimalen Gittergröße angepasst werden und wie sich dadurch die Ergebnisse verbessern. Auch für Werte von $k < 200$ übersteigen die durchschnittlichen Mehrkosten gegenüber der optimalen Strategie niemals 10, während die regulären Gitter in diesem Größenbereich zu Mehrkosten von bis zu über hundert führen.

Eine nähere Betrachtung des interessanten Größenbereichs (siehe Abbildung 6.13) offenbart weitere Vorteile der irregulären Gitter. Im Bereich von 500 bis 7000 Zellen erzielen die regulären und irregulären Gitter sehr ähnliche Ergebnisse. Gegenüber den Ergebnissen im ersten Experiment ist dabei die Schwankung in den Kosten bei den irregulären Gittern deutlich geringer. Ein Problem der regulären Gitter ist, dass sie zwar im Bereich von 500 bis 2000 Zellen sehr niedrige Schwankungen aufweisen, diese aber gerade im Bereich der optimalen Gittergröße, wo die absolut besten Ergebnisse erzielt werden, schon wieder deutlich ansteigen. Zudem gibt es zwischen den besten Ergebnissen mit 3600 und 8100 Gitterzellen einen Ausreißer mit besonders schlechten Ergebnissen. Diese Eigenschaften machen die Bestimmung einer optimalen Größe in der Praxis nicht leichter. Die irregulären Gitter dagegen weisen gerade im Bereich der optimalen Gittergröße die geringsten Schwankungen auf und eine zu hohe Wahl des Parameters k führt nur zu einer minimalen Verschlechterung gegenüber der optimalen

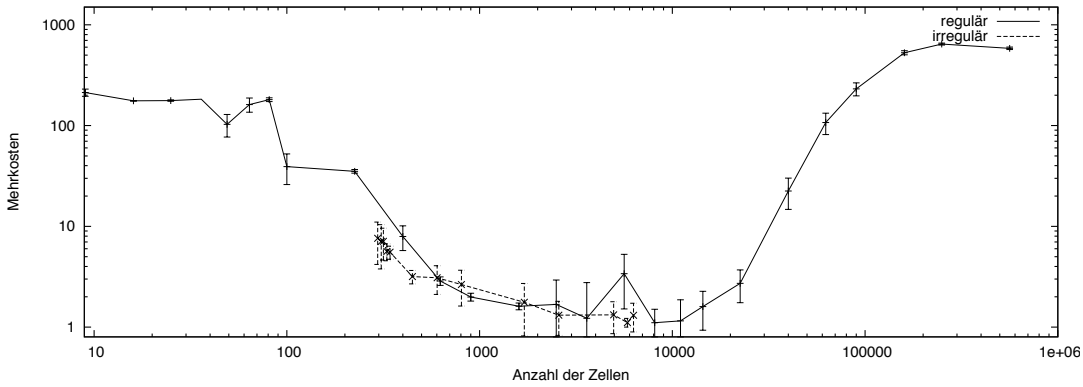


Abbildung 6.12: Mehrkosten gegenüber der optimalen Strategie bei Verwendung unterschiedlicher Gittergrößen bei Anwendung von ClusterRL auf einen Satz von 30 000 Transitionen. Aufgetragen sind bei den irregulären Gittern nicht die initialen, sondern die finalen Gittergrößen nach der automatischen Anpassung. Logarithmische Skalen.

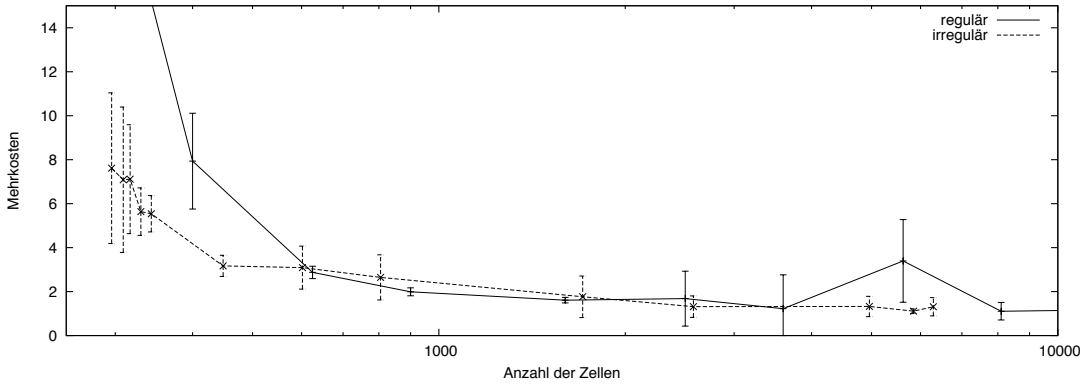


Abbildung 6.13: Ausschnitt aus Abbildung 6.12. Logarithmische x-Achse.

Gittergröße. Der Unterschied der Schwankungen zwischen regulären und irregulären Gittern bei der optimalen Gittergröße ist in Abbildung 6.14 gut zu erkennen.

Diskussion Mit den beiden Heuristiken zur Ausdünnung und Vergrößerung der Gitter stehen leistungsfähige Trainingsvarianten zur Verfügung, die es erlauben, die Stabilität der mittels ClusterRL erzielten Ergebnisse zu steigern und ihre Abhängigkeit von der Wahl der initialen Gittergröße k zu mindern. Während die automatische Ausdünnung der Gitterzellen zu große Gitter verkleinert und die mit ihnen erzielten Ergebnisse verbessert, behebt das automatische Hinzufügen von Prototypen Probleme kleiner Gitter, bei denen es sonst häufig zu Verdeckungen wichtiger Zustände – zum Beispiel der Terminalzustände – und damit zu falschen Kostenschätzungen kommt. In der Kombination ergänzen sich die Stärken der beiden Verfahren, so dass eine deutliche Verbesserung der Ergebnisse über den gesamten Wertebereich von k hinweg beobachtet werden kann.

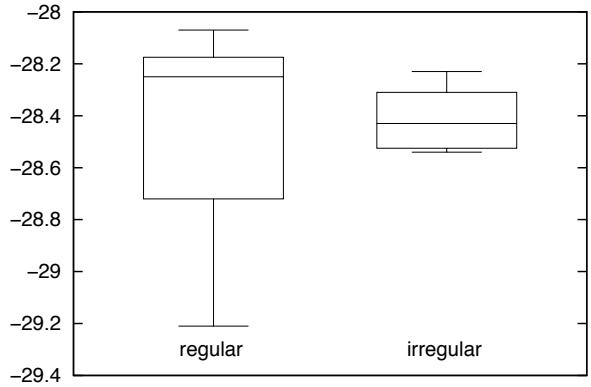


Abbildung 6.14: Vergleich der Mehrkosten bei optimaler Gittergröße auf 30 000 Transitionen. Die mittels ClusterRL erzielten Ergebnisse sind zwar im Mittel etwas schlechter, aber deutlich stabiler.

Selbst bei der Wahl viel zu kleiner Gittergrößen mit unter 100 Prototypen lassen sich bereits gute Ergebnisse erzielen, die um Größenordnungen besser als die Ergebnisse der regulären Gitter mit fester Struktur sind. Gleiches gilt für den gegensätzlichen Fehler; die Auswahl zu großer Gitterstrukturen ist praktisch nicht möglich, da die Gitter erfolgreich ausgedünnt werden. Während die sinnvolle Anzahl der Prototypen bei den irregulären Gittern generell schon durch die Anzahl der Datenpunkte auf natürliche Weise nach oben beschränkt ist, existiert bei den regulären Gittern keine solch einfach zu ermittelnde Obergrenze, was die Gefahr der Wahl zu großer Gitter birgt.

6.4.4 Lernen auf Mannigfaltigkeit

Im Rahmen der bisherigen Experimente ließen sich als Stärken der irregulären Gitterapproximatoren insbesondere die Leistungsfähigkeit im Falle von wenigen Trainingsdaten und eine geringe Abhängigkeit vom einzigen Parameter, der initialen Anzahl von Prototypen, belegen. Eine dritte wichtige Stärke der Gitter zeigt sich, wenn der Zustandsraum in einem Lernproblem –anders als beim Mountain-Car – nur ungleichmäßig abgedeckt wird und die auftretenden Zustände auf einer Mannigfaltigkeit in einem höherdimensionalen Raum liegen – die "inhärente" Dimensionalität des Systems also niedriger als die Anzahl der Dimensionen der Zustandsrepräsentation ist. Der Verdeutlichung dieser Stärke dient ein leicht verändertes Experiment mit dem Mountain-Car, bei dem das ursprüngliche System samt Startzuständen, Zielzuständen und Dynamik unverändert bleibt, die Zustände aber nicht mehr direkt beobachtbar sind, sondern in einen dreidimensionalen Beobachtungsraum eingebettet werden.

Die Zustände $s = (x, \dot{x})$ des Mountain-Cars werden unter Anwendung der nachfolgenden Abbildungsvorschrift auf die Mantelfläche eines Kreiszylinders in $\mathbb{R}^3$ projiziert:

$$f : \mathbb{R}^2 \mapsto \mathbb{R}^3 \text{ mit}$$
$$o = f(s) = (\sin x', \cos x', \dot{x}) \ ,$$

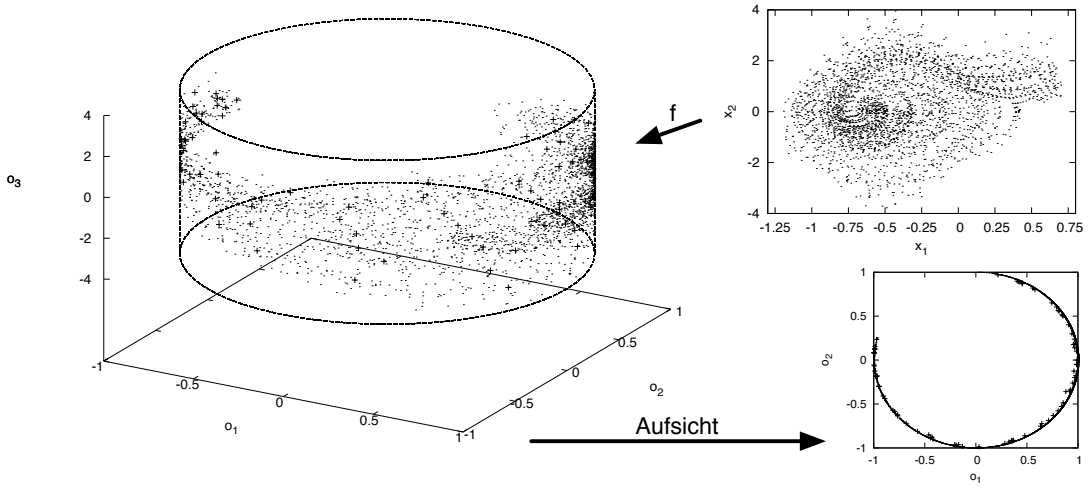


Abbildung 6.15: Projektion der Zustände des Mountain-Cars auf die Mantelfläche eines Zylinders in $\mathbb{R}^3$.

wobei $x' = 2\pi \cdot (x - posmin)/(posmax - posmin)$ die auf einen Wertebereich von $[0, 2\pi]$ normierten Werte von x annimmt. Die Lage der ursprünglichen Daten in diesem Observationsraum ist in Abbildung 6.15 zu sehen. Wichtig ist anzumerken, dass unter dieser Abbildungsvorschrift keine der beiden ersten Dimensionen des Observationsraums irrelevant ist und einfach vom Agenten ignoriert werden kann. Die Dimensionen sind linear unabhängig. Auf verschieden große Sätze von Übergängen in diesem System wurde dann der ClusterRL-Ansatz angewendet und unter den in Abschnitt 6.4.2 genannten Kriterien mit den regulären Gittern verglichen.

Das zugrunde liegende Problem ist aber durch diese Einbettung prinzipiell nicht schwieriger geworden, sofern der Lerner sich an die veränderte Lage der Daten gut anpassen kann. Bei den irregulären Gittern ist dies offensichtlich der Fall, denn bereits mit nur 3000 Übergängen werden gute Strategien erlernt, die hinsichtlich der Kosten nicht signifikant von den Strategien im $\mathbb{R}^2$ abweichen. Mit Hilfe der Tabellen ist es nicht möglich, auf dieser Datenmenge eine vergleichbar gute Strategie zu erlernen. Erst bei einer Verfügbarkeit von größeren Datensätzen nähert sich die Qualität der auf Tabellen erlernten Strategien den Ergebnissen im $\mathbb{R}^2$ an (siehe Tabelle 6.3).

6.4.5 Zusammenfassung der Evaluation

Der Vergleich des ClusterRL-Verfahrens mit irregulären Gitterapproximatoren gegenüber der Verwendung regulärer Gitterapproximatoren zeigt deutlich die Vorteile der irregulären Gitter auf. Sie sind in der Lage, ihre Struktur an die unterschiedlichen Größen der Datensätze anzupassen und dadurch insbesondere auf den kleinen Datenmengen bessere Ergebnisse zu ermöglichen. Aber auch bei Datenmengen, die groß genug zur ver-

Tabelle 6.3: Ergebnisse zum Mountain-Car in $\mathbb{R}^3$: Übersicht über die mit optimaler Auflösung erreichten durchschnittlichen Pfadkosten, den Anteil der ins Ziel führenden Testtrajektorien für reguläre und irreguläre Gitter auf verschieden großen Datensätzen (Anzahl der enthaltenen Übergänge bzw. Trajektorien in den ersten beiden Spalten).

	Strategie				Zellen		
	Kosten		Zielführend		Gesamt		Belegt
Überg.	irreg	reg	irreg	reg	irreg	reg	reg
3000	31.48	45.19	99.92%	99.85%	643	216	63
10000	29.19	43.03	99.95%	98.84%	2096	27000	1158
100000	27.50	27.97	99.99%	99.98%	3710	125000	3816

gleichbar erfolgreichen Anwendung regulärer Gitter sind, bieten die irregulären Gitter den Vorteil, dass die mit ihnen erzielten Ergebnisse wesentlich weniger abhängig von der Wahl der optimalen Gittergröße sind. Dies ist ein entscheidender Vorteil bei der geplanten Anwendung des ClusterRL-Verfahrens innerhalb von DFQ, wo die Gittergröße nicht vorab manuell bestimmt, sondern automatisch an den jeweiligen Merkmalsraum angepasst werden muss. Die im ersten Experiment beobachteten Schwankungen der Ergebnisse können durch die Trainingsvarianten mit automatischer Ausdünnung und Hinzunahme von Prototypen um die Terminalzustände ausgeglichen werden. Besonders deutlich kommen die Vorteile der irregulären Gitterapproximatoren zum Tragen, wenn in höher dimensionalen Räumen gelernt wird, in denen die Daten auf einer Mannigfaltigkeit liegen.

6.5 Deep Fitted Q-Iteration mit Clustern: DFQ-C

Mit dem entwickelten ClusterRL-Verfahren steht nun das letzte noch fehlende Werkzeug zur Verfügung, das die für einen Einsatz in DFQ gestellten Bedingungen an die Stabilität, die Anpassungsfähigkeit an die Lage der Daten und die Unabhängigkeit von manuell zu optimierenden Parametern erfüllt. Das Training der irregulären Gitter kann leicht in DFQ integriert werden. Statt bereits vor dem Start des Lernvorgangs einen konkreten Funktionsapproximator festzulegen, wird die Struktur eines Gitterapproximators erst beim Aufruf der inneren Schleife an Hand der vorliegenden Daten bestimmt und dann zur Approximation der Q-Funktion eingesetzt. Da es sich hier um den zentralen in dieser Arbeit entwickelten Algorithmus handelt, werden die einzelnen Schritte des DFQ-Algorithmus mit Clustern ausführlich aufgeführt, bevor er im folgenden Kapitel 7 auf einigen Beispielproblemen erprobt wird.

Der DFQ Variante mit Clustern muss ein Autoencoder (Netztopologie) samt Werten für die in Abschnitt 5.2.4 genannten Parameter übergeben werden. Außerdem müssen der initiale Q-Wert q^0 und die anfängliche Anzahl der Prototypen k_{init} des Gitterapproximators spezifiziert werden. Falls das Gitter ausgedünnt werden soll, wird auch ein Wert für den Parameter *min* benötigt.

A. Initialisierung Setzen eines Schleifenzählers auf Null $k \leftarrow 0$. Setzen eines Interaktionszählers auf Null $p \leftarrow 0$. Erzeugung einer initialen Explorationsstrategie $\pi^0 : Z \mapsto A$ und eines initialen Encoders ENC : $O \mapsto_{W^0} Z$. Beginn mit einer leeren Menge von Übergängen $\mathcal{F}_\mathcal{O} = \emptyset$

B. Exploration Episodische Interaktion mit dem System über eine vorgegebene maximale Episodenlänge oder bis ein Terminalzustand erreicht ist. Durchführung folgender Schritte in jedem Zeitschritt t für jede von der Umgebung kommende Beobachtung o_t eines nicht-Terminalzustands:

1. Berechnung des Merkmalsvektors $z_t = \text{ENC}(o_t; W^k)$
2. Auswahl und Anwendung der Aktion nach $a_t \leftarrow \pi^k(z_t)$
3. Sobald Rückmeldung von Umgebung, Abspeichern des vollständigen Überganges $\mathcal{F}_O \leftarrow \mathcal{F}_O \cup (o_t, a_t, r_{t+1}, o_{t+1})$ und Erhöhung des Transitionszählers $p \leftarrow p + 1$ mit jeder gespeicherten Transition

C. Encoder lernen Training der Encodergewichte W^{k+1} in einem neuen tiefen Autoencoder DEC(ENC($\cdot\,; W^{k+1}); W$) auf den p in $\mathcal{F}_O$ enthaltenen Observationen (Details in Kapitel 5). Ableitung des neuen Encoders ENC($\cdot\,; W^{k+1}$) und Erhöhung des Episodenzählers $k \leftarrow k + 1$

D. Merkmalsvektoren berechnen Anwendung des Encoders ENC($\cdot\,; W^k$) auf alle p Viertupel $(o_t, a_t, r_{t+1}.o_{t+1}) \in \mathcal{F}_O$, um sie in den Merkmalsraum Z zu überführen und so die Menge $\mathcal{F}_Z = \{(z_t, a_t, r_{t+1}, z_{t+1}) |\ t = 1, \ldots, p\}$ mit $z_t = \text{ENC}(o_t; W^k)$ zu erhalten.

E. Innere Schleife: ClusterRL Aufruf des ClusterRL-Algorithmus (Details siehe Abschnitt 6.3) mit der Menge $\mathcal{F}_Z$ der Übergänge im Merkmalsraum, dem initialen Q-Wert $\bar{q}^0$, der initialen Anzahl Prototypen k_{init} und min.

1. Gitter vorbereiten Durchführung des ClusterRL-Verfahrens wie im Abschnitt 6.3 dargestellt. Der k-means Algorithmus (siehe Abschnitt 6.3.1) wird auf den in $\mathcal{F}_Z$ enthaltenen Merkmalsvektoren ausgeführt. Mittels des initialen Q-Wertes $\bar{q}^0$ werden anschließend alle an den Prototypen gespeicherten Q-Werte initialisiert.

2. Dynamisches Programmieren Durchführung eines Schritts dynamischen Programmierens und Berechnung einer Trainingsmenge $\mathcal{P}^{i+1} = \{(z_t, a_t; \bar{q}_t^{i+1}) | t = 1, ..., p\}$ mit $\bar{q}_t^{i+1} = r_{t+1} + \gamma \max_{a' \in A} \hat{Q}^i(z_{t+1}, a')$ bzw. $\bar{q}_t^{i+1} = r_{t+1}$ für Übergänge in absorbierende Terminalzustände. Der Wert für $\hat{Q}^i(z_{t+1}, a')$ wird entsprechend Abschnitt 6.3.3 auf dem irregulären Gitter bestimmt.

3. Abspeichern Abspeichern der Trainingsdaten $\mathcal{P}^{i+1}$ im Gitterapproximator (siehe Abschnitt 6.3.2), um die $i + 1$-te approximierte Q-Funktion $\hat{Q}^{i+1}$ zu erhalten.

4. Innere Schleife Erhöhung des Schleifenzählers $i \leftarrow i + 1$ und Fortfahren mit Schritt E2, solange der Bellman-Fehler über einer vorgegeben, sehr kleinen Schranke liegt. Andernfalls Rückgabe des berechneten Fixpunktes $\bar{Q}^k$.

F. Äußere Schleife Abbruch und Rückgabe der finalen Approximation $\bar{Q}^k$ und des zugehörigen Encoders ENC$(o; W^k)$, wenn Kriterium erreicht oder Ableiten einer neuen Explorationsstrategie π^k von der Approximation $\bar{Q}^k$ (z.B. mittels ϵ-greedy-Auswertung) und Fortfahren mit Schritt B.

Dieses Verfahren wird in der Folge als DFQ-C bezeichnet, um die Verwendung des Clusteranalyseverfahrens bei der Erzeugung der irregulären Gitterstruktur des eingesetzten Approximators deutlich zu machen. Das Verfahren ist voll kompatibel mit den in Kapitel 3 vorgestellten DFQ-Varianten. Soll darüber hinaus auch die Lage der Prototypen über einen Generationswechsel der Encoder hinweg beibehalten werden, ohne noch einmal den k-means Algorithmus anzuwenden, können sie mit Hilfe des Decoders $\mathrm{DEC}(\,\cdot\,; W^{k-1})$ und Encoder $\mathrm{ENC}(\,\cdot\,; W^{k})$ in den neuen Merkmalsraum übertragen werden (siehe Abschnitt 3.5).

6.6 Zusammenfassung

In diesem Kapitel wurde die Verwendung irregulärer Gitterapproximatoren als Basis einer batch RL-Variante vorgeschlagen, um ein effizientes und vor allen Dingen stabiles Lernen in den automatisch erzeugten Zustandsräumen zu erlauben. Die Gitterstrukturen können mittels einfacher Verfahren der Clusteranalyse von den Beobachtungen abgeleitet werden. Es wurden zwei Heuristiken zur automatischen Anpassung der Anzahl der Gitterzellen an die Eigenschaften des Lernproblems vorgeschlagen, die helfen, die durch die automatische Erzeugung der Gitterstrukturen entstehende Varianz in den Ergebnissen und die Abhängigkeit von einer gut gewählten Anzahl von Gitterzellen zu senken. Im empirischen Vergleich zu den regulären, festen Gittern konnten drei wesentliche Stärken der irregulären Gitter identifiziert werden:

- Insbesondere bei einer geringen Menge von Daten besitzen die irregulären Gitter Vorteile bei der Approximation der optimalen Wertfunktion und führen schneller zu guten Strategien als die starren Gitter.
- Auch bei großen Datenmengen bieten die irregulären Gitterapproximatoren den Vorteil der größeren Unabhängigkeit von der genauen Anzahl der eingesetzten Zellen.
- Der für die Anwendung in DFQ maßgebliche Vorteil ist die Eigenschaft der irregulären Gitter, sich leicht an die Lage der Daten in Teilregionen höherdimensionaler Räume anzupassen.

Hinsichtlich der Kombination mit den durch das tiefe Lernen erzeugten Merkmalsräumen ist gerade die letztgenannte Eigenschaft der irregulären Gitter eine große Hilfe. In den automatisch erzeugten Merkmalsräumen kommt es oft vor, dass die Merkmalsvektoren in einigen Bereichen dichter beieinander liegen als in anderen Bereichen. Gerade wenn die Zieldimensionalität nicht richtig oder aus Sicherheitsgründen absichtlich zu hoch gewählt wurde (siehe Beispiel in Abschnitt 7.3), liegen die eingebetteten Daten häufig in einer unbekannten, zufälligen Mannigfaltigkeit des Raumes. Diese Situation stellt für die irregulären Gitter kein Problem dar und muss, wie im Versuch in Abschnitt 6.4.4 beobachtet, nicht zu einem schlechteren Lernergebnis führen.

Die Idee, Algorithmen aus der Clusteranalyse zur Zustandsdiskretisierung im approximativen Reinforcement Lernen zu verwenden, ist aus den genannten Gründen ei-

gentlich naheliegend. Allerdings wurden mehrere Probleme genannt, die zu instabilen Lernvorgängen bei Verwendung von online Lernverfahren wie der Vektorquantisierung führen können. Nur unter der hier erstmalig beschriebenen Kombination mit batch RL-Verfahren wie [39, 79, 113] können die Clusteranalyseverfahren so eingesetzt werden, dass stabile Lernvorgänge möglich sind. Diese Abhängigkeit und die erst kurze Verfügbarkeit der batch Verfahren mag Grund dafür sein, dass diese Verfahren bisher keine allzugroße Aufmerksamkeit erhalten haben. Mit den überlegenen Approximationsergebnissen, der geringeren Abhängigkeit vom Auflösungsparameter k_{init} und der vergleichbaren Stabilität und Handhabbarkeit wurden einige Gründe dafür genannt, warum die mittels k-means, NG oder SOM angepassten irregulären Gitter die regulären Gitter als Allzweckwaffe im batch RL ersetzen könnten.

7

Empirische Evaluation

In den Kapiteln 5 und 6 wurden die zentralen Komponenten des DFQ-Algorithmus bereits individuell evaluiert. In diesem Kapitel wird der DFQ-Algorithmus nun als Ganzes auf drei verschiedene Lernprobleme mit synthetischen und realen Bildern angewendet. Eine ausführliche, vergleichende Auswertung wird auf dem Grid-World-Problem mit synthetischen Bildern vorgenommen, da in dieser rein simulierten Umgebung alle Einflüsse unter eigener Kontrolle und für statistische Auswertungen notwendige Wiederholungen in angemessener Rechenzeit und ohne menschliche Überwachung durchführbar sind. Anschließend wird über die Lösung der gleichen Problemstellung mit einer natürlichen Bildformation der wesentliche Schritt hin zu den realen Systemen vollzogen bevor abschließend an Hand der Carrerabahn aufgezeigt wird, wie auch auf komplexen, dynamischen Systemen erfolgreich mit DFQ gelernt werden kann.

7.1 Grid-World mit synthetischen Bildern

Das einfachste der untersuchten Problemstellungen ist das bereits in Abschnitt 5.3.1 vorgestellte Grid-World-Problem mit synthetischen Bildern. Es wird hier in zwei Varianten verwendet, eine vereinfachte Version mit "diskreten" Agentenpositionen und ohne Bildrauschen und die bereits beschriebene Version mit kontinuierlichen Positionen und künstlichem Rauschen. Vor den ausführlichen Auswertungen auf der kontinuierlichen Problemstellung wird zunächst ein einzelnes Experiment auf der vereinfachten Version durchgeführt – einerseits, um die prinzipielle Funktionsfähigkeit des DFQ-Algorithmus zu belegen und anderseits, um die kaum zu unterschätzende Bedeutung des Bildrauschens zu verdeutlichen, das – wie bereits diskutiert – in bisherigen Publikationen zum Reinforcement Lernen direkt auf Bildern ignoriert wurde.

7.1.1 Machbarkeitsnachweis ohne Rauschen

Für den Machbarkeitsnachweis wird das Grid-World-Problem wie folgt abgewandelt. Während der Aufbau, die Aktionen und die Übergangsfunktion der Grid-World unverändert bleiben, wird die kontinuierliche Welt in $1m \times 1m$ große Zellen diskretisiert. Der

Agent wird ausschließlich im Mittelpunkt dieser Zellen ausgesetzt und kann sich so nur von Zellmittelpunkt zu Zellmittelpunkt bewegen, so dass eine diskrete Welt mit einer endlichen Menge von nur 31 nicht von Wänden blockierten Zuständen entsteht (siehe Abbildung 7.1 links). Als zweite Vereinfachung werden die synthetisierten Bilder auch nicht künstlich verrauscht, so dass es in der Summe nur 31 verschiedene Observationen für die 31 nicht beobachtbaren Systemzustände gibt.

Die Lernaufgabe für den Agenten besteht in dieser Problemstellung darin, die 31 hochdimensionalen Observationen zu verarbeiten und eine optimale Strategie – entspricht dem schnellsten Weg von jeder Zelle aus ins Ziel – zu erlernen. Die Aufgabe, den absorbierenden Zielzustand zu erreichen, wird als kürzestes Pfad Problem modelliert, mit Kosten in Höhe von $c = 1$ ("Belohnung" $r = -1$) für jeden Übergang, der außerhalb und $c = 0$ für jeden Übergang, der innerhalb des absorbierenden Zielzustandes endet. Der Erwartungswert für die akkumulierten Pfadkosten eines auf einem zufällig ausgewählten Feld ausgesetzten Agenten ist unter Verwendung der optimalen Strategie 5.1. Während des Lernvorgangs wird der Agent von zufälligen Positionen außerhalb des kleinen Zielraums gestartet. Die Episoden werden abgebrochen, wenn der Agent entweder das Ziel betritt oder es nicht innerhalb von 20 Schritten erreicht.

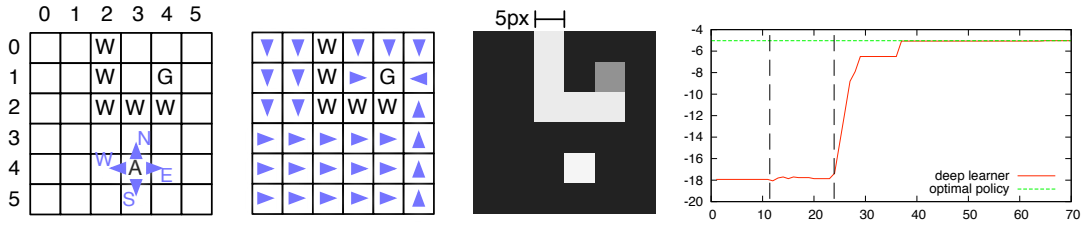


Abbildung 7.1: Zusammenfassung des Experiments auf der vereinfachten Grid-World. a) Die Welt besteht aus 31 nicht blockierten Zuständen. b) Eine optimale Strategie. c) Observationen werden ohne Rauschen synthetisiert. d) Lernkurve des DFQ-Algorithmus mit einem hochauflösenden regulären Gitterapproximator.

Es wird eine Netztopologie mit zweidimensionaler Kodierungsschicht und 11 Schichten mit 900-900-484-225-121-113-57-29-15-8-2 Neuronen im Encoderteil des Neztes verwendet, wobei die Neuronen der ersten vier versteckten Schichten in rezeptiven Feldern der Größe 9×9-mit der Vorgängerschicht verbunden sind. Die über 350 000 Verbindungsgewichte des Autoencoders werden nach 200 zufällig vom Agenten gesammelten Übergängen zum ersten Mal trainiert. Unter Verwendung eines sehr feinen regulären Gitterapproximators mit 500×500 Einträgen konvergiert der DFQ-Algorithmus innerhalb von 65 Episoden auf die optimale Strategie (siehe Abbildung 7.1, rechts). Bis zu diesem Zeitpunkt werden zwei Autoencoder trainiert und ca. 690 Zustandsübergänge eingesammelt.

7.1.2 Bedeutung von Störungen im Bildformationsprozess

Zwar gelingt es, in diesem Experiment mit DFQ in kurzer Zeit die optimale Strategie zu erlernen, aber dennoch ist die Aussagekraft für reale Anwendungen sehr beschränkt. Denn wie schon zuvor in den Experimenten von Gordon [51], Ernst [40] und Jodogne [69] ist das Lernproblem wegen des völlig deterministischen Bildformationsprozesses und der eins-zu-eins Beziehung zwischen Observationen und Systemzuständen ungleich einfacher zu lösen, als jede reale Aufgabenstellung mit einer rauschbehafteten Bildformation. Selbst ein zufallsinitialisierter, flacher Encoder wäre in der Lage, für die 31 verschiedenen Observationen in diesem Experiment unterschiedliche Merkmalsvektoren zu erzeugen. Dadurch würden die 31 Systemzustände eindeutig durch die Merkmalsvektoren identifiziert. Es wäre völlig unabhängig von ihrer Lage im Merkmalsraum möglich, die unterschiedlichen Merkmalsvektoren mit einem sehr feinen Gitter oder eine Hash-Tabelle auswendig zu lernen. Da immer wieder dieselben Beobachtungen auftreten, ist eine sinnvolle Anordnung der Merkmalsvektoren im Merkmalsraum und eine Generalisierung überhaupt nicht nötig.

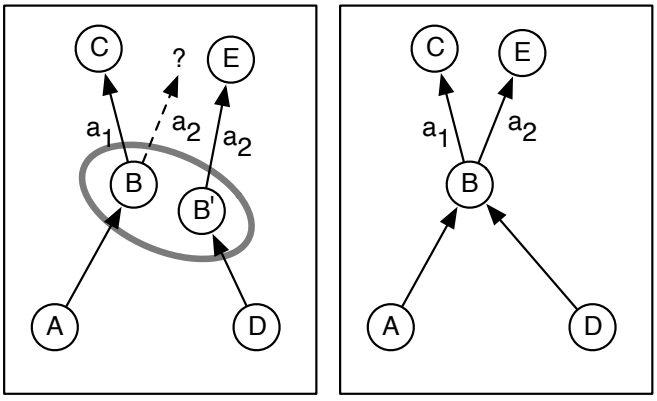


Abbildung 7.2: Bedeutung von Rauschen für das approximative Reinforcement Lernen. Kreise markieren getätigte Beobachtungen im Observationsraum, Pfeile visualisieren die beobachteten Übergänge von einer Observation zur nächsten entlang zweier Trajektorien (A,B,C und D,B',E bzw. D,B,E). In einem mit Rauschen behafteten, kontinuierlichen Prozess (links) müssen benachbarte Observationen herangezogen werden, um die Übergangskosten nicht gewählter Aktionen schätzen zu können, während sie in einem diskreten Prozess ohne Rauschen (rechts) durch neuerlichen Besuch des Zustands direkt beobachtet werden können.

Darüber hinaus kommt dem Rauschen im realen Bildformationsprozess (und anderen realen Prozessen) hinsichtlich approximativen RLs eine ganz grundsätzliche Bedeutung zu, deren Weglassen eine unzulässige Vereinfachung darstellt, wenn Aussagen zu approximativen Algorithmen gemacht werden sollen. Soll etwa in einem wertfunktionsbasierten Ansatz die Schätzung für den Q-Wert der bei Observation A gewählten Aktion (siehe Abbildung 7.2) aktualisiert werden, kann dies – wie in Kapitel 4 bereits formal eingeführt – unter Anwendung modellfreien Q-Lernens nicht allein an Hand der

Trajektorie (A, B, C) erfolgen. Vielmehr muss für die Aktualisierung sowohl der Wert der in B gewählten Aktion a_1 als auch der Wert der in B nicht gewählten Aktion a_2 herangezogen werden. In einem realen, kontinuierlichen System ist es allein aufgrund des Rauschens praktisch unmöglich, B ein zweites Mal zu besuchen, um dort auch Aktion a_2 ausprobieren zu können und den fehlenden Übergang zu beobachten[1]. Um dennoch den Wert von a_2 in B schätzen zu können, muss ein Übergang von einer anderen, nahe gelegenen Observation B' in einer zweiten Trajektorie herangezogen werden. Liegt wie im gezeichneten Beispiel den Observationen B und B' tatsächlich derselbe Zustand zugrunde, entsteht durch die Verwendung der benachbarten Observation kein Fehler in der Schätzung. Die Frage, welche Observationen sich erfolgreich zusammenfassen lassen und wo genau im Observations- bzw. Zustandsraum die Grenzen um ähnliche Zustände zu ziehen sind, ist ein Kernproblem im approximativen, übergangsbasierten Reinforcement Lernen. Fällt nun diese ganz zentrale Schwierigkeit wie bei Ernst und Jodonge einfach weg, weil die wenigen möglichen Zustände immer die gleichen, rausch- und störungsfreien Beobachtungen produzieren, kann der die Beobachtung B produzierende Zustand beliebig oft besucht werden, um alle Aktionen auszuprobieren (siehe Abbildung 7.2 rechts). So entsteht ein viel einfacher zu lösendes (diskretes) Problem, dessen Schwierigkeitsgrad nicht mit der Lösung eines kontinuierlichen Problems, das den Einsatz approximativer Algorithmen zwingend erfordert, vergleichbar ist.

Die im vorhergenden Abschnitt und in [40, 51, 69] verwendeten Experimente sind also nicht nur ungeeignet, die Robustheit der erlernten Strategie gegen natürliches Bildrauschen zu testen, sondern sind aufgrund der fehlenden Störeinflüsse und der geringen Zahl verschiedener Bilder auch durchweg ungeeignet, die eingesetzten approximativen RL-Verfahren auf die in der Realität nötige Generalisierungsfähigkeit hin zu erproben. Ein wichtiger Aspekt der approximativen Reinforcement Lernalgorithmen wird durch die Verwendung weniger, quasi diskreter Zustände vollständig ausgeblendet. Eine aussagekräftige Simulation des visuellen Reinforcement Lernproblems muss vielmehr für wiederkehrende Zustände, zum Beispiel durch Einführung künstlicher Störeinflüsse, viele unterschiedliche Observationen synthetisieren. Diese Kritik zielt zunächst gar nicht auf die Leistungsfähigkeit der vorgeschlagenen Lernmethoden ab, sondern richtet sich gegen die in den Simulationen in [40, 69] getroffenen, zu starken Vereinfachungen und zeigt die große Lücke auf, die deshalb noch zwischen den ersten Erfolgen in den (unzureichenden) Simulationen und einem erfolgreichen Einsatz an realen Systemen klafft. Um die Leistungsfähigkeit von DFQ und die Anwendungsmöglichkeit auch auf reale Anwendungen zu zeigen, ist demnach die Durchführung von Experimenten mit realistischeren, verrauschten Observationen im folgenden Abschnitt und später mit realen, von einer Kamera aufgezeichneten Bildern von grundsätzlicher Bedeutung.

[1] Dieses Problem tritt in kontinuierlichen Problemen auch ganz ohne Rauschen auf, wenn die einzelnen Trajektorien von unterschiedlichen Positionen gestartet werden.

7.1.3 Grid-World mit Rauschen

Nach dem erfoglreichen Abschluss des Machbarkeitsnachweises wird der DFQ Algorithmus nun auf das kontinuierliche Grid-World-Problem mit Bildrauschen – $\sigma = 0.1$ in allen Versuchen – angewendet. Der Agent kann in diesem Experiment jede nicht durch Wände belegte kontinuierliche Position $s \in [0, 6)^2$ im nicht beobachtbaren Zustandsraum einnehmen und sein Mittelpunkt kann in jedem der 775 freien Pixel der synthetisierten Bilder auftauchen. Aufgrund des Bildrauschens wiederholt sich in diesem Experiment selbst für identische Positionen s keine einzige Observation. Hier kommt jetzt erstmalig DFQ-C zum Einsatz.

Vorgehensweise Die Topologie des eingesetzten 8-schichtigen Encoders (Autoencoder mit 15 Schichten) ist identisch zu den in Kapitel 5 erfolgreich getesteten Netzen mit Faltungskernen: 5×5 Neuronen große Faltungskerne in den ersten beiden versteckten Schichten, danach schichtenweise voll verbunden und je Dimension halbierter Schichtgröße ab der zweiten versteckten Schicht (siehe Abbildung 7.3). Die irregulären Gitter werden mit den Varianten zur Ausdünnung und Anreicherung um die absorbierenden Zielzustände bestimmt (siehe Abschnitt 6.3), wobei die Anzahl der initialen Prototypen auf $k_{\text{init}} = 100$ gesetzt wird. Eine von den beschriebenen Standardeinstellungen abweichende Feineinstellung der Netztopologie und Parameter ist nicht nötig, da DFQ sich auch hier robust gegenüber einer Änderung der Parameter – z.B. Größe der rezeptiven Felder und Anzahl von initialen Zentroiden – verhält (siehe auch Kapitel 5 und 6).

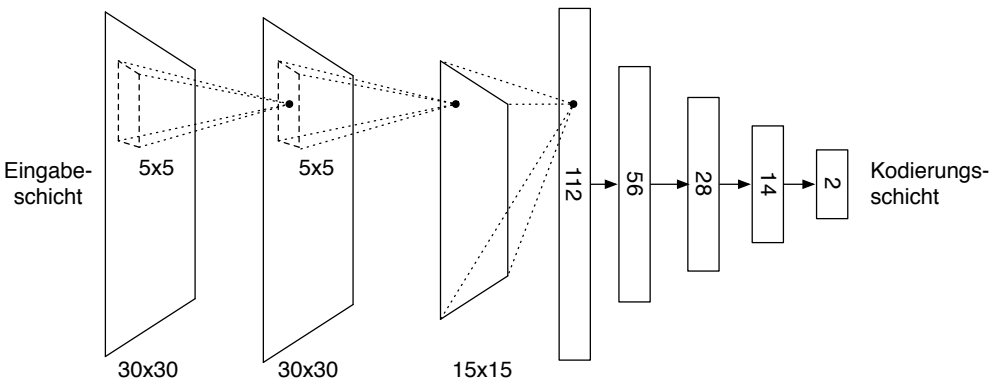


Abbildung 7.3: Netztopologie der im Experiment eingesetzten Encoder.

Gestartet wird der Agent in jeder Episode an einer zufällig gewählten, kontinuierlichen Position $s = [0, 6)^2$ außerhalb des Zielraumes, so dass der schmale Durchgang vom Agenten selbständig entdeckt werden muss, bevor er das Ziel erreichen kann. Der Agent bewegt sich in jedem Zeitschritt $1m$ in eine von vier Richtungen, wobei wie im Versuch zuvor für jeden Schritt Kosten in Höhe von $c = 1$ entstehen, sofern der Schritt nicht im $1m^2$ großen Zielbereich endet. Nach spätestens 20 Schritten wird die Episode beendet. Der erste Encoder wird in diesen und in allen weiteren Experimenten nach

dem Einsammeln von 100 Beobachtungen trainiert.

Zur Exploration verwendet der Agent eine "bipolare" Explorationsstrategie, bei der nach jeder Episode zwischen gewöhnlicher "pessimistischer" – die Kosten bisher nicht besuchter Aktionen werden überschätzt – ϵ-greedy Exploration mit $\epsilon = 0.3$ und "optimistischer" – die Kosten neuer Aktionen werden unterschätzt – Exploration [150] ohne Zufallseinfluss gewechselt wird. Die zweite Strategie ist eine einfache, globale, zählerbasierte Explorationsstrategie [162, 163]. Das stetige Umschalten zwischen beiden Explorationsarten ist durch die Verwendung von batch RL leicht möglich und basiert auf unterschiedlichen initalen Q-Werten $\bar{q}^0$ – einmal über den maximal, einmal unter den minimal möglichen Kosten – bei der Vorbereitung der Gitter. Die bipolare Explorationsstrategie stellt einen robusten Kompromiss zwischen aggressiver Exploration noch nicht besuchter Bereiche unter der optimistischen Explorationsstrategie – ähnlich einer Breitensuche – und vorsichtiger Exploration um die bisher besten bekannten Pfade unter der pessimistischen ϵ-greedy Strategie dar.

Prinzipiell ist die Lösung des Grid-World-Problems auch mit gewöhnlicher ϵ-greedy Exploration möglich. Mit dieser bipolaren Explorationsstrategie wurden aber gerade bei den Grid-World-Problemen vielversprechende Erfahrungen gesammelt und Beschleunigungen des Lernvorganges bis zum Faktor 2 beobachtet. Aufgrund der damit verbundenen Verringerung des Interaktionsaufwandes wurde die ungewöhnliche bipolare Explorationsstrategie der ansonsten verwendeten ϵ-greedy Exploration in diesem Experiment vorgezogen, da die eingesparte Rechenzeit eine ausführlichere statistische Untersuchung mit mehr Wiederholungen ermöglichte.

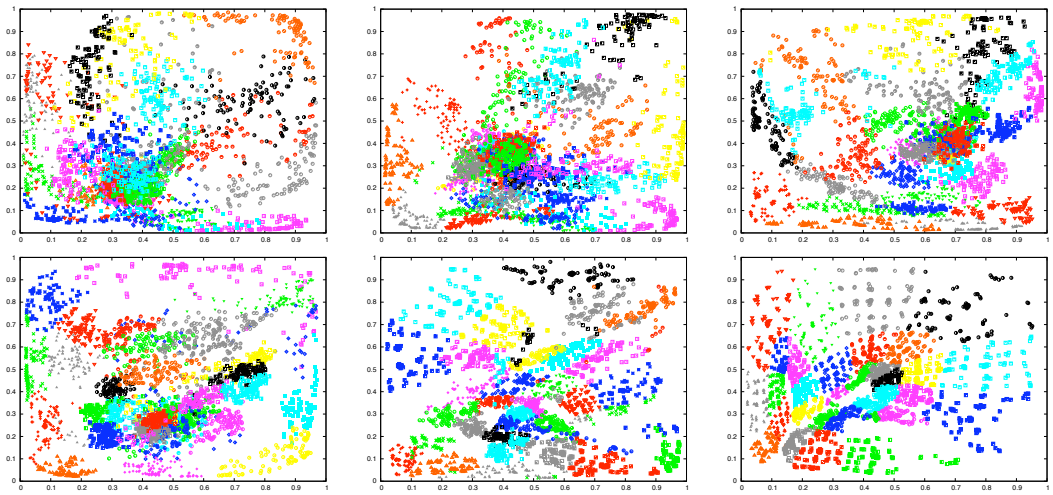


Abbildung 7.4: Von links nach rechts: Merkmalsräume der Encodergenerationen eins bis sechs in einem Lernversuch.

Lernvorgang Während der Lerndauer von 500 Episoden[1] sammelt der Agent insgesamt 5681 Übergänge ein und trainiert insgesamt sechs Autoencoder nach 110, 222, 464, 939, 1896 und 3795 Schritten. Die Merkmalsräume der sechs Encodergenerationen sind in Abbildung 7.4 dargestellt. Es ist eine deutliche Entwicklung hin zu einer saubereren Trennung der Bilder entsprechend der Position des Agenten sichtbar. Ab Episode 363 erreichen die nach jeder Episode abgeleiteten und getesteten gierigen Strategien immer den Zielzustand von allen Testfällen. Bereits ab Episode 47 wird der Zielzustand von mehr als der Hälfte der Startzustände erreicht und ab Episode 105 erreichen einzelne Strategien das Ziel von allen getesteten Startzuständen innerhalb der erlaubten 20 Schritte. Die beste Strategie nach 500 Episoden verursachte im Mittel von allen getesteten Startpositionen Kosten in Höhe von 5.18, was bis auf unter einen Zehntelschritt an der optimalen Strategie mit mittleren Kosten von 5.1 liegt. In Kapitel 5 war es selbst bei einer perfekten, uniformen Verteilung der Beobachtungen nicht möglich, auf den Bildern der Grid-World eine Klassifikationsrate von 100% zu erzielen. Wenn dies auch im überwachten Lernen selbst bei gleichmäßiger Abdeckung nicht möglich ist, ist es nicht verwunderlich, dass es auch hier zu einzelnen Verwechslungen kommt.

Im Lernverlauf sind einige Schwankungen bei der Güte aufeinander folgender Strategien zu beobachten (siehe Abbildung 7.7). Dies liegt vornehmlich an der nach jeder Episode neu vorgenommenen Clusterung des Merkmalsraums und den unterschiedlichen Resultaten. In den letzten 100 Episoden liegen die Strategien aber selbst im Mittel nur 0.23 Schritte über dem Resultat der optimalen Strategie. Die Standardabweichung über die von den letzen 100 erlernten Strategien erzielten Kosten hält sich mit 0.103 ebenfalls in engen Grenzen, so dass die Auswahl einer guten Strategie nicht schwer fällt. Selbst die absolut schlechteste dieser letzten 100 Strategien liegt mit 5.58 weniger als einen halben Schritt von der optimalen Strategie entfernt.

Erlernte Wertfunktion In Abbildung 7.5 ist eine von der erlernten Q-Funktion abgeleitete Zustandswertfunktion dargestellt. Zum Vergleich ist rechts die optimale Zustandswertfunktion im Zustandsraum abgebildet. Der Verlauf der zum Ziel hin abnehmenden Kosten wird gut approximiert. An Hand des Voronoi-Diagramms ist zu erkennen, dass die Auflösung des irregulären Gitters um den Zielzustand höher als nötig ist. Dies rührt von der insbesonders zum Anfang hilfreichen Initialisierung zusätzlicher Clusterzentren auf allen Übergängen in den Zielzustand her. Eventuell kann hier durch eine stärkere Ausdünnung im späteren Lernverlauf schneller gelernt werden.

Statistische Auswertung Eine fünfmalige Wiederholung des Lernversuchs erlaubt eine Untersuchung der auftretenden Schwankungen. Nach einer unterschiedlich verlaufenden Anfangsphase verlaufen die Lernversuche nach einigen Encodergenerationen sehr ähnlich und es treten nur noch geringe Schwankungen in den erzielten Ergebnissen auf (siehe Abbildung 7.6). Drei individuelle Lernverläufe sind in Abbildung 7.7 zusammen mit den erlernten Merkmalsräumen beispielhaft abgebildet. Auch wenn zwischen

[1] Dieses Lernversuch dauert mit Auswertung bei Verwendung von 10 parallelen Threads über 8 Stunden.

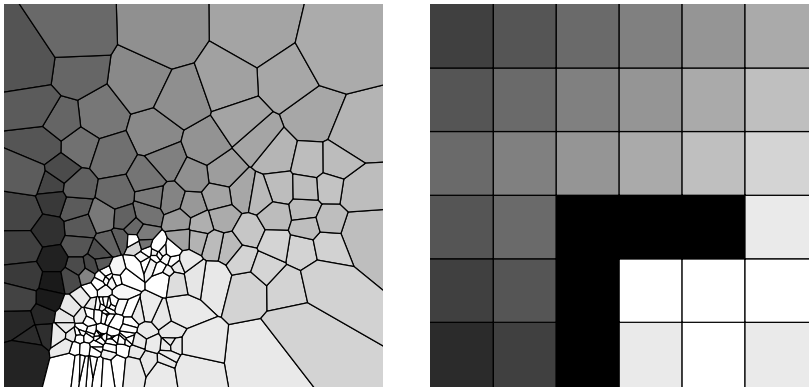


Abbildung 7.5: Darstellung der erlernten Q-Funktion als Zustandswertfunktion im Merkmalsraum (Voronoi-Diagramm links). Zum Vergleich die optimale Wertfunktion im Zustandsraum (Diagramm rechts). Dunklere Schattierungen bedeuten höhere Kosten.

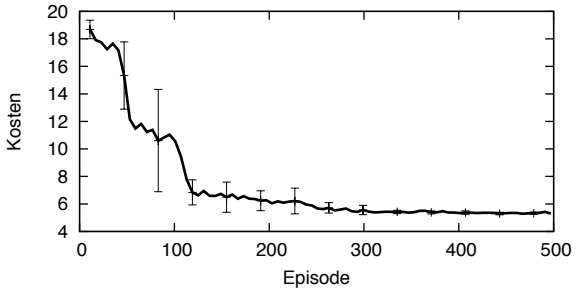


Abbildung 7.6: Durchschnittlicher Lernverlauf mit Standardabweichungen.

den Anordnungen der Zustände deutliche Unterschiede bestehen, werden in allen drei Merkmalsräumen die Observationen der einzelnen Agentenpositionen gut voneinander getrennt. Dies unterstreicht noch einmal die Fähigkeit des tiefen Lernens, unabhängig von den Ausgangsgewichten zu einer sinnvollen, separierenden Einbettung der Observationen in in den Merkmalsraum zu gelangen. Obwohl der dritte abgebildete Merkmalsraum rein optisch am geordnetsten aussieht, wird die absolut beste Strategie in einem der anderen Lernversuche erzielt (Kosten von 5.17).

Trotz der sichtbaren Unterschiede im Lernverlauf und bei den konstruierten Einbettungen, werden in allen Lernversuchen mit Hilfe des Encoders der sechsten Generation vergleichbar gute Strategien erzielt. So weichen die Kosten der jeweils besten Strategie in den einzelnen Lernversuchen nur geringfügig voneinander ab (siehe Tabelle 7.1, Zeile DFQ-C mit Weight Sharing, Spalte "beste Strategie"). Ähnliches gilt für die absolut schlechtesten Strategien im späteren Lernverlauf von Episode 400–500 (Spalte "schlechteste Strategie"). Auch die Schwankungen innerhalb der einzelnen Lernversuche bewegen sich im späten Lernverlauf in einem engen Rahmen: Zwischen den durchschnittlich verursachten Kosten während der letzten 100 Episoden des besten und des schlechtesten

Tabelle 7.1: Vergleich der Kosten und Schwankungen bei Verwendung regulärer (DFQ-R) und irregulärer Gitter (DFQ-C). Spalte 3 und 4: Mittelere Kosten und Standardabweichung der besten und schlechtesten Strategie (ab Episode 400) in 5 unabhängigen Lernversuchen. Spalte 5 und 6: Schwankung der Strategiegüte von Episode zu Episode (ab Episode 400) beschränkt auf den besten und den schlechtesten der 5 unabhängigen Lernversuche. Spalte WS: Mit oder ohne Weight Sharing.

Methode	WS	⌀ Kosten		⌀ Kosten	
		beste Strategie	schlechtes. Strategie	bester Lernverlauf	schlechtes. Lernverlauf
DFQ-R	•	5.81 ± 0.770	6.21 ± 1.242	5.42 ± 0.006	7.74 ± 0.446
DFQ-C	•	5.20 ± 0.044	6.00 ± 0.613	5.29 ± 0.063	5.38 ± 0.212
DFQ-C	∘	5.74 ± 0.350	7.55 ± 0.900	5.66 ± 0.180	7.04 ± 0.535

der fünf Lernversuche liegen weniger als $\frac{1}{10}$ Schritte und auch die Schwankungen unterscheiden sich nur um den Bruchteil eines Schrittes (Spalten "bester / schlechtester Lernverlauf").

Vergleich mit regulären Gittern Zum Vergleich wird dasselbe Experiment mit einem regulären Gitterapproximator (bezeichnet als DFQ-R) vorgenommen. Selbst bei optimaler Wahl der Gittergröße (Versuchsreihe zur Bestimmung der besten Gittergröße siehe Abbildung 7.8) sind die erzielten Ergebnisse schlechter als bei Verwendung der irregulären Gitter (siehe Tabelle 7.1). Die erzielten Strategien sind im Mittel deutlich schlechter als bei Verwendung der Gitter. Gleichzeitig sind die Unterschiede zwischen den einzelnen Lernvorgängen größer und stärker abhängig vom Verlauf der Exploration, was sich in der wesentlich höheren Standardabweichung bemerkbar macht. Tatsächlich wird selbst im schlechtesten Lernverlauf mit irregulärem Gitter immer noch eine Strategie erlernt, die besser ist als jede Strategie in allen fünf Lernversuchen mit regulärem Gitter (5.28 gegenüber 5.37). Zudem entfällt bei Verwendung von DFQ-C die aufwändige Bestimmung der optimalen Gittergröße.

Vergleich mit rezeptiven Feldern In einer letzten Versuchsreihe soll der Einfluss des Weight Sharings innerhalb der Faltungskerne der Encoder untersucht werden. Zu diesem Zweck werden die Faltungskerne durch einfache rezeptive Felder ohne geteilte Gewichte ersetzt. Alle anderen Parameter bleiben unverändert. Die unter Verwendung von DFQ-C mit rezeptiven Feldern erzielten Ergebnisse sind insgesamt deutlich schlechter (siehe Tabelle 7.1, letzte Zeile) als die mit Faltungskernen erzielten Ergebnisse.

7.1.4 Zusammenfassung

Zusammenfassend lässt sich feststellen, dass DFQ-C in der Grid-World nahe an der optimalen Strategie liegende, stabile Ergebnisse erzielt und sich die Anwendung der irregulären Gitter (DFQ-C) den regulären Gittern (DFQ-R) als deutlich überlegen erweist – sowohl hinsichtlich der Güte erlernter Strategien als auch hinsichtlich der geringeren Schwankungen der Ergebnisse.

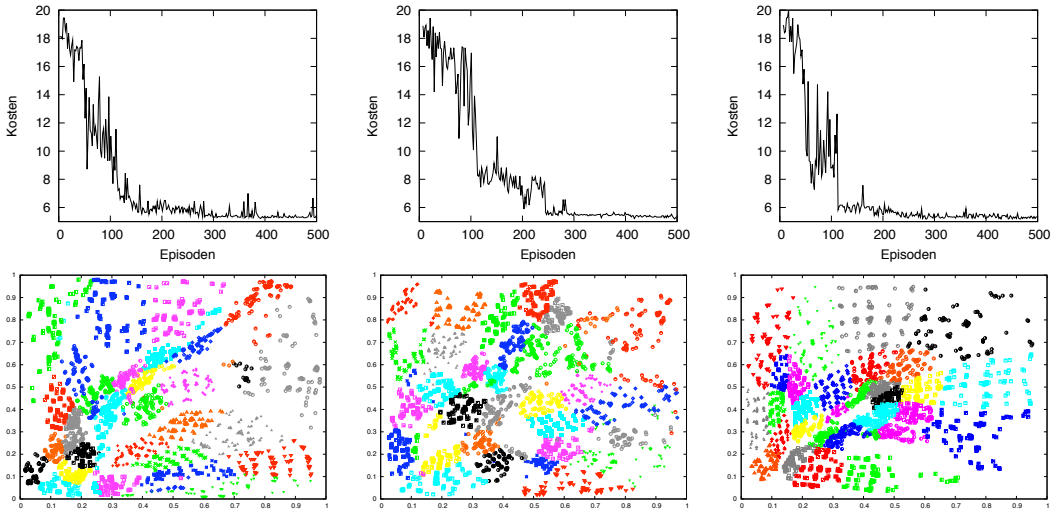


Abbildung 7.7: Lernverläufe in drei unabhängigen Wiederholungen des Grid-World-Experiments (oben) und zugehörige, finale Merkmalsräume (darunter). Aufgeführt sind die durchschnittlichen Kosten über alle möglichen Startfelder.

7.2 Grid-World mit realer Bildformation

Der Grid-World-Versuch wird nun mit Hilfe einer realen Bildformation wiederholt. Zu diesem Zweck wird das Spielfeld samt Agenten auf einen Bildschirm gezeichnet und mit Hilfe einer digitalen Videokamera abgefilmt. Bei diesem Vorgehen bleibt der Vorteil simulierter Systeme, dass die internen Zustände bekannt sind und der Versuchsablauf unter voller Kontrolle ist, weiter bestehen. Gleichzeitig wird mit der Bildformation aber der in dieser Arbeit zentrale Aspekt in die Realität verlegt und nicht mehr simuliert. In diesem Versuch müssen die tiefen Autoencoder eine sinnvolle Repräsentation aus den realen Bildern erzeugen und dazu mit natürlichem Bildrauschen, Pixelaliasing, einer leichten Kissenverzerrung (siehe Abbildung 7.9) und einer ungleichmäßigen Ausleuchtung des Bildbereichs umgehen.

Die von der Kamera aufgezeichneten Bilder werden auf 1/4 QVGA-Größe verkleinert (80×60) Pixel. Es wird weiterhin die bewährte Netzarchitektur aus dem simulierten Versuch verwendet. Aufgrund der über fünfmal größeren Bilder – 4800 statt 900 Pixel – ergibt sich allerdings ein noch tieferes Netz mit 10 Schichten. Während das Training tiefer Autoencoder dieser Größenordnung auch mit der parallelen Implementierung einige Stunden dauert, lassen sich einzelne Bilder aber in Echtzeit durch die BLAS-beschleunigten Encoder propagieren, so dass das Experiment mit 15Hz durchgeführt werden kann. Der Agent wird wie im vorhergehenden simulierten Experiment an zufällig ausgewählten Positionen $s \in [0, 6)^2$ ausgesetzt. Episoden werden wiederum nach maximal 20 Schritten abgebrochen. Die erlernten Strategien werden nach jeder zweiten Episode von 31 festen Startzuständen getestet, die den Mittelpunkten der be-

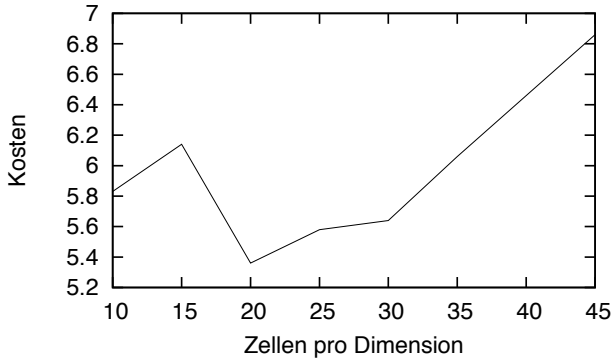


Abbildung 7.8: Durchschnittliche Kosten der besten innerhalb von 500 Episoden gefundenen Strategie bei Verwendung unterschiedlich großer regulärer Gitter.

reits mehrfach verwendeten $1m \times 1m$ Zellen entsprechen.

Verwendet wird in diesem Versuch ein relativ kleines initiales Gitter mit nur $k_{\text{init}} = 100$ Zellen. Nach 47 Episoden hat der Agent mittels DFQ-C bereits eine Strategie erlernt, die ihn in 90% der Fälle bis ins Ziel führte (siehe Abbildung 7.10). Im Schnitt verursacht diese Strategie Kosten von 9. Nach ca. 300 Episoden erreicht der Agent stabile Strategien, die nur noch selten Schwanken. Die absolut beste erlernte Strategie erzielt recht eindrucksvolle durchschnittliche Kosten in Höhe von 6, was innerhalb eines Schrittes von der optimalen Strategie entfernt liegt. Nach dem Training des fünften Encoders nach 420 Episoden verschlechtern sich die Strategien allerdings wieder um bis zu 1.5 zusätzliche Schritte. Dies ist ein Beispiel dafür, dass sich die erlernten Merkmalsräume von einer auf die andere Generation auch verschlechtern können. Im Hinblick auf praktische Anwendungen erscheint eine Überprüfung und Auswahl der besten erlernten Strategie ("Policy Inspection") daher generell sinnvoll.

7.3 Carrerabahn: Lernen an dynamischen Systemen

Im Carrerabahn-Experiment fährt ein Wagen unter einer Kamera und muss so gesteuert werden, dass einerseits möglichst gute Rundenzeiten erzielt werden, andererseits der Wagen aber auch nicht die Haftung verliert und von der Strecke rutscht. Die zur Verfügung stehenden Informationen sind die Kamerabilder und eine Messung darüber, ob der Wagen sich noch in der Bahn befindet oder bereits den Kontakt verloren hat. Diese Information kann z.B. über die Messung des in der Bahn fließenden Stroms gewonnen werden. Hier wird das bereits in [74] eingesetzte, auf CVTK-basierende [80] System verwendet, das auch Ground-Truth Informationen zu Auswertungszwecken liefern kann. Der Rechner gibt an die Bahn ein eindimensionales Steuersignal zurück, das die anzulegende Spannung spezifiziert. Die verwendete Strecke ist in Abbildung 7.11 dargestellt. Tatsächlich ist das Erlernen von Strategien selbst mit einer handkodierten

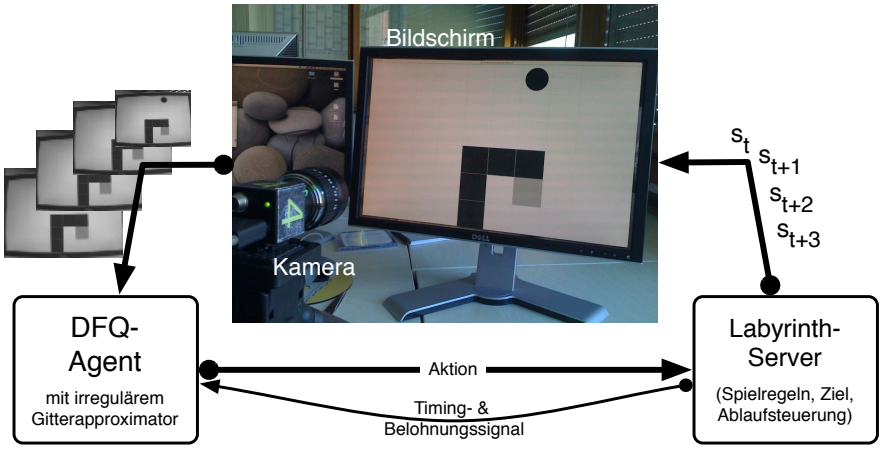


Abbildung 7.9: Versuchsaufbau des Grid-World-Experiments mit realer Bildformation.

Bildverarbeitung nicht einfach und wurde erst kürzlich beschrieben [74]. Hier soll nun mittels DFQ-C eine Strategie ohne handkodierte Bildverarbeitung direkt auf Bildern erlernt werden.

7.3.1 Modellierung

Die Carrerabahn unterscheidet sich in einigen grundsätzlichen Aspekten von den bisher untersuchten Problemstellungen. Sie ist dynamisch, Übergänge sind stochastisch und Reaktionen des Wagens sind im Grenzbereich schwer vorherzusagen. Um zu verdeutlichen, wo die Schwierigkeiten bei diesem dynamischen System im einzelnen liegen, werden in der Folge die bisher angestellten und im Rahmen einer noch laufenden Masterarbeit weiter verfolgten Überlegungen diskutiert, die nötig sind, um an diesem System erfolgreich lernen zu können. Da der zentrale Aspekt dieser Arbeit das Erlernen von Strategien auf Bildern und nicht so sehr der Umgang mit dynamischen Systemen ist, wird die Diskussion an dieser Stelle kurz gehalten.

Zustandsbeschreibung Eine markovsche Repräsentation des Systemzustands ist für die erfolgreiche Anwendung von wertfunktionsbasierten Reinforcement Lernverfahren wie DFQ auch auf realen Systemen zwingend erforderlich (siehe Kapitel 2). Bei der Steuerung des Wagens spielt nicht nur die Position, sondern auch die aktuelle Geschwindigkeit eine entscheidende Rolle. Sie kann aber nicht direkt aus einem Bild bestimmt werden, so dass ein tiefer Encoder keine Möglichkeit hat, diese Information im Merkmalsraum zu kodieren. Hier wird nun die Idee verfolgt, die Geschwindigkeit aus der Differenz zwischen aktuellem und vorhergehendem Merkmalsvektor zu berechnen. Gleichzeitig treten – wie bei realen, kamerabasierten Systemen unvermeidbar – recht hohe Verzögerungen zwischen Messung und Wirkung der ausgewählten Aktion auf. Wenn dem Rechner ein Bild zur Verfügung steht, ist der Wagen schon ein Stück weiter ge-

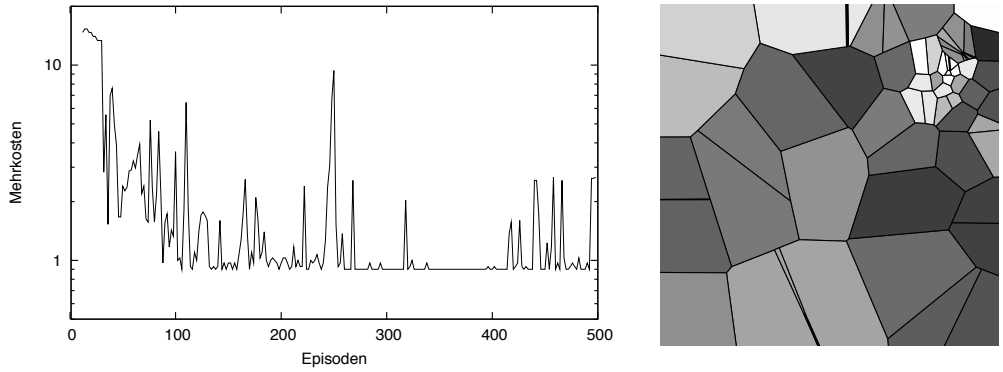


Abbildung 7.10: Links: Von der erlernten gierigen Strategie gegenüber der optimalen Strategie durchschnittlich erzielte Mehrkosten (logarithmische y-Achse). Die besten Strategien sind im Bereich zwischen 300 und 400 Episoden zu finden. Rechts: Erlernte Partitionierung und approximierte Zustandswertfunktion der ersten, von allen Testzuständen zielführenden Strategie im zweidimensionalen Merkmalsraum. Dunklere Schattierungen bedeuten höhere erwartete Kosten.

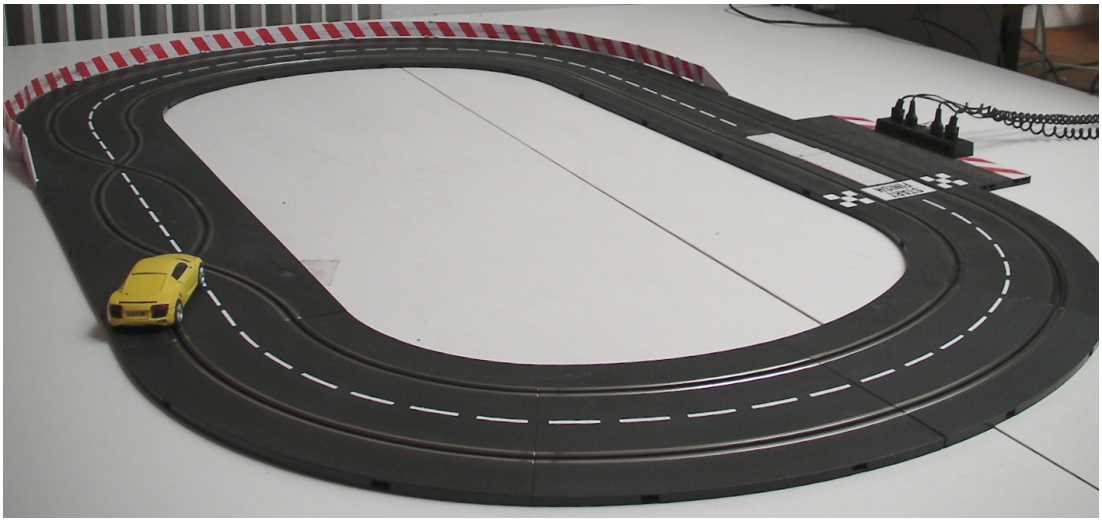


Abbildung 7.11: Carrerabahn mit Doppelschikane (direkt vor dem Wagen), Hochgeschwindigkeitskurve mit Bande (oben) und Kurve ohne Bande.

fahren[1] und bis die auf Basis dieses Bildes gewählte Aktion wirkt, vergeht weitere Zeit. Die totale Verzögerung über die vollständige Schleife – von einer initialen Messung, über Auswahl und Anlegen des Steuersignals bis hin zur ersten Messung, in der die

[1] Nach eigenen Messungen mit digitalen Industriekameras nach dem IIDC-Standard sind mindestens zwei weitere Taktzyklen seit der Aufzeichnung vergangen. Das entspricht bei 60Hz mindestens 33ms.

ersten Auswirkungen des Steuersignals sichtbar sind – wurde experimentell auf 15 bis 16 Kamerabilder, also ca. 250–280ms, bestimmt.

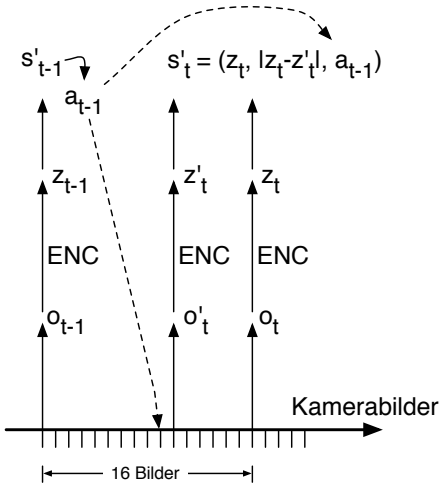


Abbildung 7.12: Konstruktion des angereicherten Zustandes s'_t aus observierten Bildern und der vorhergehenden Aktion.

Mit diesen Verzögerungen kann auf verschiedene Weise umgegangen werden. Einerseits kann ein prädiktives Modell analytisch bestimmt oder erlernt werden, um den tatsächlich für die Entscheidung relevanten Zustand aus den veralteten Messungen vorherzusagen. Diese Vorgehensweise wurde mit [46, 85, 86] für die Versuche in [132, 134] verfolgt. Eine andere Möglichkeit besteht darin, ältere Messungen und Aktionen in den Zustand mit aufzunehmen. Dadurch entsteht ein Markov-Entscheidungsprozess höherer Ordnung, der sich wiederum mit den wertfunktionsbasierten RL-Methoden lösen lässt. Die hier verfolgte, in Abbildung 7.12 dargestellte Alternative besteht darin, die Regelfrequenz so weit abzusenken, dass die letzte Änderung am Steuersignal bereits Eingang in die nächste Messung, auf deren Basis die folgende Entscheidung getroffen wird, genommen hat [105]. Dabei wird die von der Kamera vorgegebene Taktung verwendet, wobei die Zeitschritte $t = 1, 2, \ldots$ 16 Kamerabilder entsprechend der gemessenen Verzögerung auseinander liegen.

Aufgrund dieser Überlegungen muss außerdem eine Zustandsbeschreibung $s'_t = (z_t, \Delta z)$ gewählt werden, die nicht nur aus dem aktuellen Merkmalsvektor z_t, sondern auch aus der Veränderung Δz gegenüber einem früheren Merkmalsvektor z'_t besteht. Die Differenz Δz wird aber nicht einfach aus den Merkmalsvektoren z_t und z_{t-1} berechnet. Vielmehr wird ein möglichst nah an z_t liegendes Kamerabild zur Differenzbildung verwendet, das gerade noch eine zuverlässige Unterscheidung zwischen verschiedenen Geschwindigkeiten zulässt. Die in diesem Sinne optimale Differenz kann empirisch bestimmt werden und beträgt 5 Frames. Da diese Beschreibung immer noch nicht ganz markov ist – in die Differenzbildung gehen die Einflüsse der zuletzt gewählten Aktion nur zum Teil ein und es kann an der Position und Geschwindigkeit allein nicht erkannt

werden, ob die Antriebsräder die Traktion verloren haben oder der Wagen gar bereits durch die Kurve driftet – wird zudem die zuletzt gewählte Aktion als weitere Dimension in die Zustandsbeschreibung $s'_t = (z_t, \Delta z, a_{t-1})$ aufgenommen.

Aktionen, Belohnung und Diskontierung Dem DFQ-Agenten stehen vier die angelegte Spannung beeinflussende Aktionen zur Auswahl: Die Aktion 80 führt zu einer niedrigen aber auch bei Daueranwendung sicheren Endgeschwindigkeit, mit der der Wagen nicht von der Strecke fliegt. Die Aktion 120 führt zu einer schnelleren Fahrt, liegt aber knapp über der an allen Stellen der Strecke sicheren Geschwindigkeit. Aktion 200 beschleunigt den Wagen maximal und führt mit Sicherheit binnen weniger Zeitschritte, lange vor Erreichen der maximalen Endgeschwindigkeit, zum Abflug des Wagens von der Strecke. Aktion 0 bremst den Wagen aktiv durch Kurzschließen der Schleifkontakte. Die Belohnungen werden an die Aktionen geknüpft: Bremsen bewirkt eine Belohnung von 0, die anderen Aktionen erhalten 90, 120 bzw. 150 als sofortige Belohnung. Um eine absolut sichere Strategie zu erhalten, die Unfälle unter allen Umständen zu verhindern sucht und nicht an manchen Stellen niedrige Unfallrisiken für eine höhere kurzfristige Belohnung eingeht, wird der Agent im Falle eines Unfalls mit $-1\,000\,000$ heftig bestraft.

Anders als die Grid-World lässt sich dieses System nicht auf natürliche Weise als kürzester Pfad Problem modellieren; der Wagen soll viele Runden so schnell wie möglich fahren, ohne ein Ziel zu erreichen oder irgendwo zu stoppen. Formuliert wird deshalb ein Lernproblem mit unendlichem Horizont und relativ starker Diskontierung. Der Diskontierungsfaktor wird mit $\gamma = 0.1$ so gewählt, dass der Agent tendenziell diejenige Aktion auswählt, die die höchste 1-Schritt Belohnung verspricht – der Wagen soll so schnell wie möglich fahren – dabei aber auch einige Schritte in die Zukunft voraus schaut, um einen Unfall zu vermeiden.

7.3.2 Versuchsablauf

Die Episoden dauern 80 Schritte (ca. 20 Sekunden), sofern der Wagen nicht vorher den Kontakt zur Bahn verliert. Einzelne, gute Strategien werden in einem anschließenden Testlauf über eine längere Zeit von 400 Zeitschritten verifiziert. Bei der Clusteranalyse des Merkmalsraums werden die Trainingsvarianten zur Anpassung der Zellenanzahl hier aufgrund des Fehlens absorbierender Zielzustände nicht verwendet.

Training des Autoencoders Bei der Carrerabahn können Observationen von allen Positionen des Wagens durch eine einfache Strategie mit niedriger, konstanter Geschwindigkeit gesammelt werden. Daher wird das Training des Autoencoders aufgrund der langen Dauer (> 12 Stunden) entsprechend der in Abschnitt 3.5.4 vorgestellten Variante (“Herauslösen des Encodertrainings”) vom Erlernen der Strategie entkoppelt und vor dem Start des eigentlichen Lernversuchs mit vom Agenten gesteuerter Exploration durchgeführt. Die Observationen werden auf eine Größe von 76×69 Pixeln verkleinert. Trainiert wird 3000 Bildern (bei 60 Hz inklusive weiterer Testbilder in weniger als 2 Minuten aufzuzeichnen), zum Einsatz kommt immer eine der schon bekannten generi-

schen Netzarchitekturen mit 9×9 Neuronen großen Faltungskernen in den ersten zwei versteckten Schichten.

Exploration Dieses System kennzeichnet ein sehr großer Zustandsraum, über den nur bedingt verallgemeinert werden kann: Bei welcher Geschwindigkeit der Wagen aus der Bahn rutscht, ist nicht an jeder Position gleich, sondern von dem lokalen Streckenverlauf abhängig. In der Schikane und in der Kurve ohne Bande führen schon relativ niedrige Geschwindigkeiten zum Unfall, während die Kurve mit Bande mit beliebig hoher Geschwindigkeit durchfahren werden kann, da der Wagen hier von der Bande geführt wird. Um eine hohe Belohnung zu erhalten, möchte man möglichst nah an diese Diskontinuität – auf der einen Seite gibt es eine hohe Belohnung, auf der anderen Seite gibt es mit einer leicht höheren Geschwindigkeit aber bereits die maximale Bestrafung – in der Wertfunktion heranrücken, ohne auf die gefährliche Seite zu rutschen. Die Lage der Diskontinuität müsste daher für ein optimales Ergebnis exakt bekannt sein, denn anders als zum Beispiel im Mountain-Car-Problem kann man sich von den gefährlichen Zuständen nicht fernhalten, sondern muss so nah wie möglich heran an die Diskontinuität. Insgesamt entsteht so ein beliebig hoher Explorationsaufwand[1] und ein gewissermaßen unvermeidbares Maß an Instabilität, wenn die Grenze zu den gefährlichen Geschwindigkeiten nicht genau approximiert wird. Aus diesem Grund wird ein in drei Phasen gegliederter Trainingsablauf mit unterschiedlichen Explorationsstrategien verwendet:

1. **Demonstrationsphase** In dieser Phase wird Imitationslernen verwendet, um dem Agenten eine initiale, sichere Strategie entsprechend der Idee zum Einbringen von Vorwissen in Abschnitt 3.5.4 zu zeigen. In den ersten zwei Episoden wird ausschließlich die ungefährliche Aktion 90 von außen vorgegeben. Die dabei gesammelten Transitionen gehen gleichberechtigt in die Trainingsmenge ein und "schneiden" von Anfang an ein sicheres Band durch den Zustandsraum. Diese Phase ist nicht erforderlich für den Lernerfolg, beschleunigt den Lernvorgang allerdings deutlich.

2. **Explorationsphase** Ausgehend von der sicheren Strategie wird nun mit einer niedrigen Explorationsrate ($\epsilon = 0.1$) exploriert. Typischerweise wird der Wagen langsam schneller. Allerdings wird mit zunehmender Steigerung der Geschwindigkeit die Strategie auch unsicherer und es kommt zu mehr Unfällen.

3. **Konsolidierungsphase** In dieser Phase wird das bisher Erlernte in einer Strategie konsolidiert und stabilisiert. Dazu wird die Exploration ausgeschaltet – also rein gierig ausgewertet – und die Position der Gitterzellen fixiert. So werden langsam alle unsicheren Aktionen ausgeschlossen, die derzeit noch aufgrund bisher günstiger stochastischer Übergänge fälschlich als vielversprechend erscheinen. Gleichzeitig wird sichergestellt, dass keine Stützstellen mehr in dünn explorierte Bereiche verschoben werden.

[1] Zusätzlicher Explorationsaufwand bedeutet hier längere menschliche Beaufsichtigung, um den Wagen zurück auf die Strecke zu legen, und ist nicht wie in der Simulation kostenfrei zu haben.

Filterung Ein Problem mit dem realen System entsteht durch grobe Messfehler, die zu fehlerhaften Übergängen führen und die geschätzten Kosten verfälschen können. Hier hilft eine einfache Überwachung, die Transitionen verwirft, wenn der Kamerazyklus nicht eingehalten wird – hin und wieder kommt ein Bild z.B. durch Unterbrechungen des Prozesses zu spät an – oder eine negative Geschwindigkeit (s.u.) auftritt. Unter diesen Kriterien fallen weit weniger als 1% der Transitionen raus; aber dennoch hat sich diese Maßnahme als wichtig für den Erfolg und die Stabilität der Lernvorgänge erwiesen.

7.3.3 Merkmalsraum

Ein erstes Experiment verwendet einen zweidimensionalen Merkmalsraums (siehe Abbildung 5.16 in Kapitel 5). Um überhaupt zunächst die Tauglichkeit der erlernten Einbettung zu überprüfen, wird die Geschwindigkeit nicht im Merkmalsraum aus den Merkmalsvektoren (z_t^1, z_t^2) berechnet, sondern mit Hilfe des CVTK-basierten Objektverfolgungssystems im Bildraum gemessen und im Zustand $s'_t = (z_t^1, z_t^2, v_t, a_{t-1})$ als v verwendet. Die Kodierung der Position wird in diesem Experiment also von DFQ erlernt, während die Geschwindigkeit zunächst von außen zugeführt wird. Auf Basis dieser Zustandsrepräsentation erreichte die beste, sichere Strategie – keine Unfälle in der 400 Schritte andauernden Evaluationsphase – eine durchschnittliche sofortige Belohnung von 115.9 pro Zeitschritt (siehe Tabelle 7.2). Dies ist eine deutliche Steigerung zur in der Demonstrationsphase vorgeführten Strategie und zur Zufallsstrategie, die beide nur 90 erreichen – wenn man bei der Zufallsstrategie die Bestrafungen bei den unvermeidlichen Unfällen nicht mitrechnet. Wird die auf klassische Weise gemessene Geschwindigkeit nun durch die Differenz der erlernten Merkmalsvektoren $\Delta z_t = ||z_t - z'_t||$ ersetzt, erzielte der DFQ-Agent mit der besten, sicheren Strategie eine Belohnung von 97.9 pro Aktion.

Tabelle 7.2: Von den verschiedenen Verfahren erzielte durchschnittliche Belohnungen.

Strategie	Belohnung	Gittergrösse	Zustand
Zufall	90.0	–	
Sicher	90.0	–	
DFQ-C	115.9	400	4D: (z_1, z_2, v, a_{t-1})
DFQ-C	97.9	400	4D: $(z_1, z_2, \|\Delta z\|, a_{t-1})$
DFQ+SOM	113.8	200	3D: $(z', \|\Delta z'\|, a_{t-1})$
DFQ+SOM	120.2	800	3D: $(z', \|\Delta z'\|, a_{t-1})$

Wird die gemessene Geschwindigkeit durch die Differenz der Merkmalsvektoren $\Delta z_t = ||z_t - z'_t||$ ersetzt, erzielte der DFQ-Agent mit der besten, sicheren Strategie eine Belohnung von 97.9 pro Aktion. Dies ist zwar eine Steigerung gegenüber der sicheren, in der Demonstrationsphase vorgeführten Strategie, bleibt aber deutlich hinter den Erwartungen zurück. Als eine Ursache hat sich die Art der Differenzbildung im zweidimensionalen Merkmalsraum erwiesen. Alle in mehreren Versuchen für die Carrerabahn

erlernten, zweidimensionalen Merkmalsräume besitzen mindestens ein oder zwei Sprünge entlang des Bandes (siehe Abbildung 5.16), die zu einer Verfälschung der über den euklidischen Abstand der Merkmalsvektoren geschätzten Geschwindigkeit führen (siehe Abbildung 7.13). Auch die Windungen können bei langen Schrittlängen im Merkmalsraum zu verfälschten Ergebnissen führen. Eine statistische Auswertung in der Nähe der Sprünge zeigt, dass an diesen Stellen unterschiedliche reale Geschwindigkeiten auf Basis der berechneten Differenzen nicht mehr zweifelsfrei identifiziert werden können (siehe Abbildung 7.15). Ohne diese Unterscheidungsmöglichkeit kann an diesen Stellen aber bestenfalls eine sehr konservative Strategie erlernt werden, die mit allen Geschwindigkeiten funktioniert. Dies erklärt die schlechteren Ergebnisse im Vergleich zum Experiment mit der extern gemessenen Geschwindigkeit.

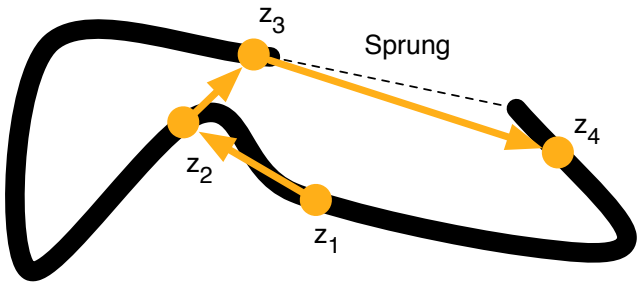


Abbildung 7.13: Schematische Darstellung der im zweidimensionalen Merkmalsraum (s. Abb. 5.16) auftretenden Probleme. Aufgrund einer "Abkürzungen" der Schleife erscheint der Übergang von z_2 zu z_3 kürzer als der Übergang von z_1 zu z_2, während der Übergang von z_3 zu z_4 aufgrund eines Sprungs im Merkmalsraum ungewöhnlich lang erscheint.

In Abbildung 5.17 ist zu erkennen, dass dieses Auseinanderreißen durch "mangelndem Platz" beim Entfalten der Repräsentation entsteht. Die Lösung besteht darin, dem Autoencoder einen zusätzlichen Freiheitsgrad einzuräumen. Bereits drei statt zwei Neuronen in der Kodierungsschicht reichen aus, um sowohl Sprünge als auch Überschneidungen entlang des Bandes sicher zu verhindern. Eine der erzeugten dreidimensionalen Einbettungen ist in Abbildung 7.14 dargestellt. Anders als es die perspektivische Darstellung des Raums vermittelt, gibt es entlang des Bandes keinerlei Überschneidungen oder Berührungen nicht benachbarter Regionen.

Statt eine weitere Dimension in die Zustandsrepräsentation $s' = (z_t^1, z_t^2, z_t^3, \Delta z_t, a_{t-1})$ für die zusätzliche Dimension z_3 des Merkmalsraums aufzunehmen und so das Lernproblem weiter zu verkomplizieren, kann in diesem Fall mittels eines SOMs leicht eine eindimensionale Kette (am Anfang und Ende verbunden) eingebettet werden (siehe Abbildung 7.14). Die dreidimensionale Position z_t im Merkmalsraum kann dann nach einer senkrechten Projektion des Merkmalsvektors auf das SOM als eindimensionale, reellwertige SOM-Koordinate $\hat{z}_t \in [0, 1]$ entlang der eingebetteten Topologie angegeben werden. Der große Vorteil ist nun, dass auch die Geschwindigkeit als Differenz entlang des SOMs berechnet werden kann und so gleichzeitig das Problem mit den scheinbaren Abkürzungen der Windungen (siehe Abbildung 7.13) behoben wird. In Abbildung 7.15 wird die

so berechnete Differenz mit der im zweidimensionalen Merkmalsraum berechneten Differenz verglichen. Es ist gut zu erkennen, wie viel besser die realen Geschwindigkeiten durch die SOM-Differenz repräsentiert werden.

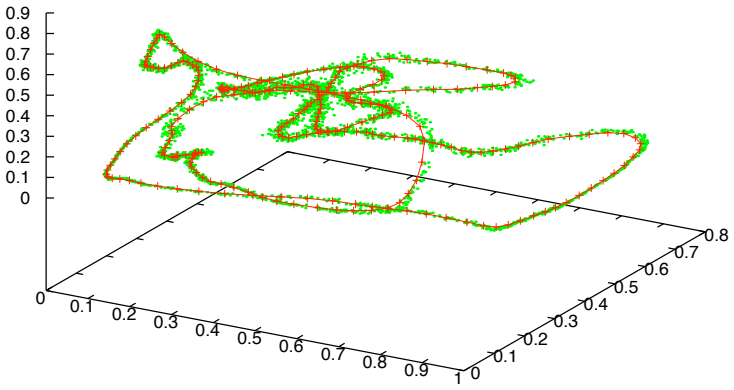


Abbildung 7.14: Erlernter dreidimensionaler Merkmalsraum mit einer in die Daten (grün) eingebetteten SOM in Form einer eindimensionalen Kette (rot).

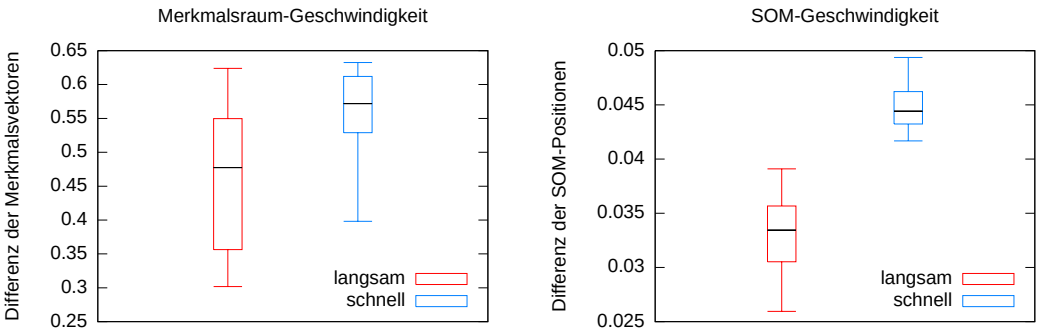


Abbildung 7.15: Vergleich der im Merkmalsraum berechneten Geschwindigkeiten nahe einer problematischen Stelle des Merkmalsraums (links) und der auf der SOM berechneten Geschwindigkeiten (rechts). Zu sehen ist eine statistische Auswertung wiederholter Geschwindigkeitsmessungen des Wagens bei dauerhafter Fahrt mit einer von zwei verschiedenen Aktionen (langsam / schnell). Die SOM-Geschwindigkeit ist eine erkennbar bessere Repräsentation der tatsächlich gefahrenen Geschwindigkeit, da die Verteilungen der zu den jeweiligen realen Geschwindigkeiten gehörenden Messungen bei der SOM deutlich weiter auseinander liegen und keine Überschneidungen aufweisen.

7.3.4 Ergebnisse: Lernverlauf und erlernte Strategie

Verwendet wird der dreidimensionale Zustand $s_t = (\hat{z}_t, \Delta\hat{z}_t, a_{t-1})$, wobei $\hat{z}_t$ die Position auf dem SOM und $\Delta\hat{z}_t$ eine Differenz auf dem SOM ist. Der Lernverlauf ist in Abbildung 7.16 dargestellt. Nach der anfänglichen Demonstrationsphase ist gut zu erkennen, wie die durchschnittliche sofortige Belohnung (ohne Berücksichtigung der Bestrafungen) auf Kosten der Sicherheit ansteigt und die Episodenlänge aufgrund von Unfällen abnimmt. Nach dem Übergang in die Konsolidierungsphase nach Episode 80 nimmt die durchschnittliche Belohnung wieder leicht ab, während die Zuverlässigkeit aber schnell ansteigt, bis der DFQ-Agent schließlich eine stabile Strategie zur Verfügung hat. Die finale Strategie mit 200 Zentroiden erzielte bei einem Test über 400 Zeitschritte eine durchschnittliche Belohnung von 113.8, was sehr nah an dem Vergleichsexperiment mit direkt gemessener, "echter" Geschwindigkeit liegt. Mit 800 Zentroiden gelingt es in einem zweiten Versuch sogar, mittels DFQ-C eine sichere Strategie zu erlernen, die diese Belohnung auf knapp über 120 pro Schritt steigert.

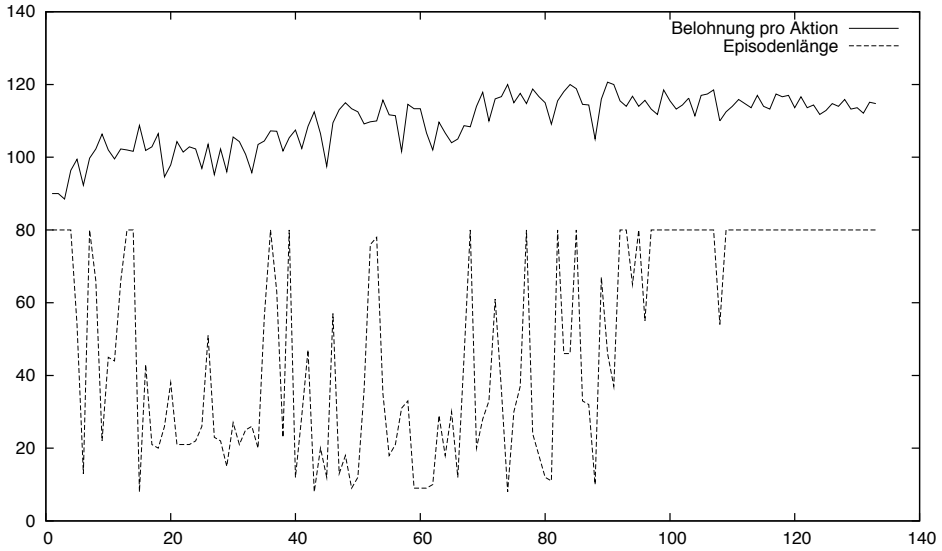


Abbildung 7.16: Lernverlauf im Carrerabahn-Experiment mit kurzer Demonstrationsphase (ersten zwei Episoden), Explorationsphase (bis Episode 80) und Konsoliedierungsphase (ab Episode 80). Dargestellt ist die erzielte Episodenlänge und die durchschnittlich pro Aktion erhaltene Belohnung ohne Berücksichtigung der Bestrafung bei Unfällen.

Die erlernte Strategie mit 200 Prototypen ist in Abbildung 6.14 visualisiert. Man erkennt gut, wie der Wagen aus der Schikane heraus maximal beschleunigt und dann zum Ende der Zielgeraden scharf vor der Kurve ohne Bande bremst, um dann mit gemäßigter Geschwindigkeit durch sie und die anschließende Schikane hindurch zu fahren.[1]

[1]Bei der Interpretation des Bildes sollte beachtet werden, dass der Agent an Hand des aktuellen Bildes eine Aktion für einen späteren Zeitpunkt wählt; der Wagen befindet sich aufgrund der Verzöge-

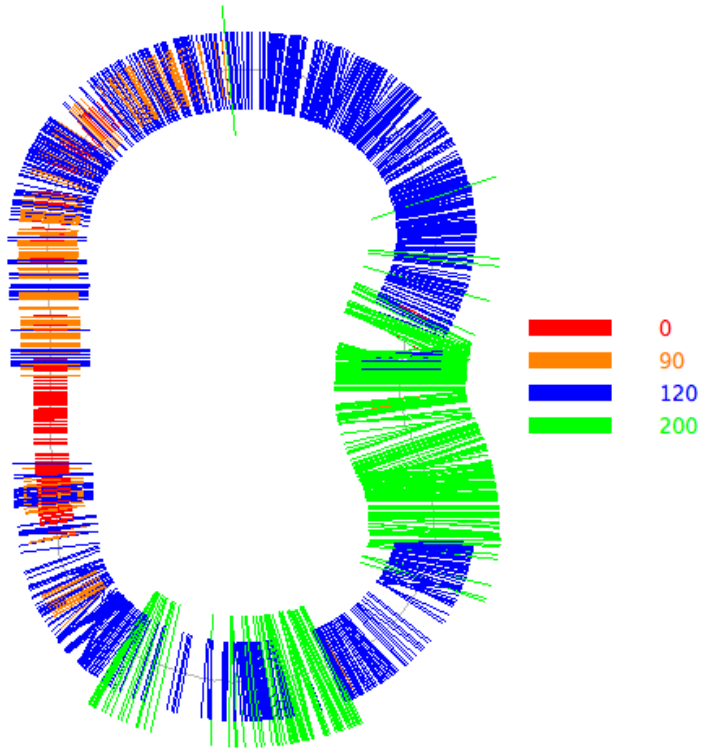


Abbildung 7.17: Visualisierung der erlernten Strategie über die Darstellung der vom Agenten gewählten Aktionen über mehrere Runden ununterbrochener Fahrt.

7.3.5 Diskussion der Ergebnisse

Mit der Carrerabahn wurde ein anspruchsvolles, dynamisches System untersucht. Mit Hilfe der bereits aus Kapitel 5 bekannten zweidimensionalen Einbettung wurden gute Strategien im Merkmalsraum erlernt. Die Differenzbildung im Merkmalsraum ist eine Methode, eine Repräsentation der Geschwindigkeiten aus den Merkmalsvektoren zu extrahieren. Im Vergleich zur Verwendung einer auf klassische Weise gemessenen Geschwindigkeit sind waren die erzielten Resultate etwas schlechter, konnten aber durch eine Einbettung der Observationen in drei Dimensionen und die Verwendung einer SOM weiter verbessert werden. Die auf dieser Bahn erzielten Ergebnisse stehen den mittels NFQ und einer klassischen Bildverarbeitung erzielten Ergebnissen [74] in nichts nach. Im Vergleich zu NFQ bieten die mittels der irregulären Gitter in der Konsolidierungsphase erlernten Strategien zudem den Vorteil, dass sie von einer Aktualisierung auf die andere sehr stabil sind und sich kaum noch verändern. Tatsächlich bietet es sich sogar

rungen schon etwas weiter vorne auf der Strecke.

an, auch in der Anwendungsphase bei ausgeschalteter Exploration weiter zu lernen, um Aktionen mit extrem selten auftauchenden kritische Übergängen im Falle eines Unfalls noch nachträglich ausschließen zu können.

Die Technik, eine SOM in einen eigentlich zu hochdimensionalen Merkmalsraum einzubetten, ist von allgemeinem Interesse und kann überall dort angewendet werden, wo einem Autoencoder absichtlich oder unabsichtlich – aufgrund der Unkenntnis der "intrinsischen" Dimension – mehr Freiheitsgrade in der Kodierungsschicht als nötig gegeben werden, aber die grundsätzliche Topologie in einem sinnvollen Zustandsraum bekannt ist. Bei der Carrerabahn war diese Topologie ein eindimensionaler Ring – der Wagen bewegt sich nur auf der durch die Führungsrille vorgegebenen Strecke – bei dem Grid-World-Beispiel würde man ein zweidimensionales Gitter vorgeben. Dieses Vorgehen ähnelt dem in [74] verwendeten, klassischen System, bei dem auch eine Streckenposition und Geschwindigkeit berechnet werden – nur, dass diese Verarbeitung hier vollständig durch den Einsatz selbstorganisierender Methoden ohne Überwachung aus den Daten erlernt wird.

7.4 Zusammenfassung

Bei allen drei Experimenten kam die gleiche, generische Netztopologie mit Faltungskernen in den beiden äußeren versteckten Schichten und mit einer Halbierung der Schichtgrößen ab der zweiten versteckten Schicht zum Einsatz. Zwar variierte die Breite der Schichten und die Tiefe der Netze in Abhängigkeit von der Anzahl der Pixel der verwendeten Observationsgröße, dennoch wurden mit Hilfe dieser Architektur in allen drei untersuchten Lernproblemen vom DFQ-C-Algorithmus überzeugende, wiederholbare Ergebnisse erzielt. Im simulierten Grid-World-Experiment wurden nahezu optimale Strategien erlernt. DFQ-C zeigte sich zudem der Verwendung regulärer Gitter mit optimaler Größe überlegen. Selbst die Varianz der Ergebnisse war bei Verwendung von DFQ-C niedriger als bei DFQ-R. Auch die Faltungskerne haben sich bewährt und zu deutlichen Verbesserung gegenüber den rezeptiven Feldern geführt. Anschließend wurde die Methode zunächst erfolgreich auf ein simuliertes Zwischenexperiment mit realer Bildformation übertragen, um dann schließlich auf das anspruchsvolle Carrerabahn-Problem angewendet zu werden. Hier wurde die volle Leistungsfähigkeit von DFQ-C sichtbar. Nur auf Basis der unvorverarbeiteten Bilder wurden sinnvolle Merkmalsräume und Strategien erlernt. Die mit einem zweidimensionalen Merkmalsraum erlernten Strategien waren bereits gut, auf einem dreidimensionalen Merkmalsraum waren sie noch deutlich besser und sind vergleichbar zu den Ergebnissen, die mit Hilfe einer klassischen Bildverarbeitung erzielt wurden [74]. Dabei benötigten die Lernversuche zwar wegen des aufwändigen Autoencodertrainings insgesamt mehr Rechenzeit, aber die Lernvorgänge selbst führen in der gleichen Größenordnung von Episoden zum Erfolg.

Die Verwendung eines höherdimensionalen Merkmalsraums als eigentlich nötig erschwert das Lernproblem nicht wie vielleicht erwartet, sondern kann sogar durch den zusätzlichen Freiheitsgrad zu Verbesserungen bei der Anordnung der Merkmalsvektoren führen. Die Daten verteilen sich nicht chaotisch im größeren Raum, sondern behalten

ihre "intrinsische" Topologie, nehmen nur eine Mannigfaltigkeit des größeren Raumes ein und formen weiterhin ein sehr enges Band, das nun aber nicht mehr so viele Überschneidungen und Sprünge aufweist. Ist die grundsätzliche Topologie wie bei der Carrerabahn bekannt, kann durch Techniken wie die Einbettung von selbstorganisierenden Karten die Dimensionalität des für den Lerner konstruierten Zustandsraums dennoch klein gehalten werden. Die so erlernten Repräsentationen besitzen einen ähnlichen Informationsgehalt wie die auf klasssische Weise gewonnene Zustandsbeschreibung.

8

Resümee

In diesem Schlussteil der Arbeit werden die wichtigsten Ergebnisse zusammengefasst, im Sinne der eingangs aufgestellten Anforderungen diskutiert und in den Forschungskontext eingeordnet. Es werden einige vielversprechende Möglichkeiten für anknüpfende Arbeiten aufgezeigt, bevor mit einem kurzen Fazit geschlossen wird.

8.1 Zusammenfassung

Zielsetzung dieser Arbeit war die Entwicklung und Untersuchung von Methoden zur Lösung visuomotorischer Lernprobleme, die es erlauben, sequentielle Entscheidungsprobleme direkt auf unverarbeiteten, ikonischen Bilddaten nur durch den Einsatz von maschinellen Lernverfahren zu lösen. Während diese Aufgabe uns Menschen spielend leicht fällt, stellt sie aktuelle Verfahren des optimierenden Lernens immer noch vor eine Reihe großer Herausforderungen.

Mit dem entwickelten DFQ-Verfahren wurde der klassische, zweistufige Lösungsansatz verfolgt, bei dem zwischen dem Wahrnehmungsteilproblem in der ersten Stufe und dem Erlernen einer Handlungsstrategie in der zweiten Stufe unterschieden wird. Anders als in bestehenden Verfahren kommen in DFQ ausschließlich unüberwachte und optimierende Lernverfahren zum Einsatz, die selbständig mit der Umgebung interagieren und ohne Eingriffe und Hilfestellungen vom Menschen auskommen. Es wurde aufgezeigt, wie sich tiefe Autoencodernetze zum unüberwachten Erlernen kompakter Merkmalsräume auf effiziente Weise mit modernen, modellfreien batch RL-Verfahren in einem neuen, integrierten Ansatz mit episodischer Exploration kombinieren lassen. Mit dem sofortigen Neuprogrammieren einer Strategie nach einem Generationswechsel, dem Übersetzen der erlernten Wertfunktionen und der Rekonstruktion der Observationen aus den Merkmalsvektoren wurden Methoden aufgezeigt, die Effizienz – vornehmlich im Sinne der Dateneffizienz, aber auch hinsichtlich der Laufzeit und Speicheranforderungen – dieser Verbindung weiter zu steigern.

Tatsächlich ist der entwickelte Ablaufrahmen auch ganz unabhängig von der hier gewählten konkreten Realisierung der beiden Stufen mit tiefen Autoencodern und Fitted Q-Iteration von Bedeutung. Ganz allgemein können die entwickelten Methoden immer

dann zum Einsatz kommen, wenn mittels speicherbasierter batch RL-Methoden auf einem sich mit der Zeit verändernden Zustandsraum gelernt werden soll. Die Möglichkeit im Lernverlauf Zustandsräume zu erweitern oder zu ändern kann selbst für ganz klassische Problemstellung ohne visuelle Perzeption interessant sein, wenn zum Beispiel aufgrund neuen Wissens oder angestoßen durch ein externes, den Lernerfolg analysierendes Modul zusätzliche Informationen oder Merkmale in die Zustandsbeschreibung aufgenommen werden sollen.

Im Kapitel 4 wurde das DFQ-Verfahren theoretisch untersucht und auf Basis der Ergebnisse von Gordon [53] sowie Ormoneit und Sen [113] nachgewiesen, dass es für diskontierte MDPs beim Einsatz geeigneter Approximationsverfahren zu einem eindeutigen Fixpunkt konvergiert. Auch wenn sich im Fall nicht diskontierter MDPs die Konvergenz genauso wie die allgemeine stochastische Konsistenz des Verfahrens nicht formal nachweisen lassen, wurden wichtige Kriterien und Heuristiken für die Stabilität auch in diesen Fällen und für die allgemeine Konsistenz des DFQ-Algorithmus formuliert. Auf Basis dieser Überlegungen wurden in Kapitel 5 die von den tiefen Autoencodern erzeugten Merkmalsräume eingehend untersucht und in Kapitel 6 eigens ein im batch RL stabiler und für den Einsatz in DFQ besonders geeigneter Funktionsapproximator entwickelt und im neuen ClusterRL-Verfahren erprobt.

Die von den Autoencodern erzeugten Merkmalsräume erwiesen sich in der empirischen Untersuchung als geeignet für den Einsatz in RL-typischen Navigationsproblemen. Mittels des entwickelten Lernverfahrens mit rezeptiven Feldern und insbesondere mit den Faltungskernen kann die Einbettung von Bildern in einen geeigneten Merkmalsraum mit nur zwei Dimensionen gelingen – selbst wenn im Vergleich zu bisher publizierten Ergebnissen nur wenige, ungleichmäßig verteilte Trainingsbilder zur Verfügung stehen. Lokale Nachbarschaften im ursprünglichen, nicht beobachtbaren Raum der Systemzustände werden dabei gut erfasst und auch die globale Topologie wird zu einem gewissen Grad in dieser unüberwachten Lernprozedur in den Merkmalsraum abgebildet. Mittels des überwachten Trainings auf eine Zielfunktion kann die Merkmalsextraktion weiter verbessert und an die speziellen Erfordernisse der Strategielernaufgabe angepasst werden.

Mit dem ClusterRL-Algorithmus wurde ein Verfahren entwickelt, dass einerseits nach den theoretischen Ergebnissen stabile Lernvorgänge ermöglicht und andererseits optimal auf den Einsatz in DFQ und den Umgang mit den automatisch erzeugten Merkmalsräumen abgestimmt ist. Mit Hilfe von batch Verfahren aus der Clusteranalyse wird die Struktur eines irregulären Gitterapproximators automatisch an die Lage der Daten (im Merkmalsraum) angepasst, ohne dabei abhängig von guten, von außen vorgegebenen Parametern zu sein. Bei der Entwicklung dieses Verfahrens wurden Probleme bestehender Verfahren aufgezeigt, die Methoden der Vektorquantisierung mit online RL-Verfahren kombinieren. Ähnliche Probleme wurden im ClusterRL-Verfahren durch den Einsatz von batch Verfahren und durch geeignete Aktualisierungsregeln und Ablaufsteuerung vermieden. Das Verfahren wurde zwar speziell für den Einsatz in DFQ entwickelt, zeigte sich aber selbst optimal eingestellten regulären Gitterapproximatoren nicht nur wegen der geringeren Abhängigkeit von den Parametern überlegen und ist

auch über diese Verwendung in DFQ hinaus anwendbar.

Für die Durchführung der Versuchsreihen und den geplanten Einsatz an realen Systemen war eine Beschleunigung bestehender Bibliotheken und echtzeitfähige Implementierung zwingend erforderlich. Auch dynamisches Programmieren auf den Transitionen, das Clusteranalyseverfahren und die Zugriffe auf die irregulären Gitter wurden parallelisiert. Besonderes Augenmerk wurde aber auf die Parallelisierung des Trainings der tiefen Autoencodernetze gelegt, da hier der größte zeitliche Zugewinn zu erzielen war. Neben der Umordnung der Gewichte und der anschließenden BLAS-Beschleunigung des Vorwärts- und Rückwärtspropagierens wurde ein Verfahren zur Beschleunigung speziell von Gradientenabstiegsverfahren mit batch Aktualisierungen wie RProp entwickelt. Durch Aufteilung der Daten und parallele Berechnung der Teilgradienten auf mehreren Netzkopien kann das Training effektiv beschleunigt werden, wobei diese Art der Parallelisierung wegen des geringeren Synchronisationsaufwandes besser skaliert als eine Parallelisierung auf Neuronenebene. Auf einem Mehrprozessorsystem konnte so eine 27-fache Beschleunigung gegenüber einer sequentiellen Verarbeitung in n++ erzielt werden, wodurch die hier beschriebenen Anwendungen überhaupt erst ermöglicht wurden.

In den in Kapitel 7 erprobten Anwendungen wurden durchweg gute Ergebnisse erzielt. Im Gird-World-Problem mit synthetischen Bildern hat sich der DFQ-C Algorithmus als den regulären Gittern überlegen erwiesen. In wiederholbaren, stabilen Lernverläufen wurden nahezu optimale Strategien erlernt. Auf dem Grid-World-Problem mit realen Bildern wurden ähnlich gute Ergebnisse erzielt. Auch bei der Steuerung der Carrerabahn, einem schwierigen, dynamischen System, konnten sehr gute Ergebnisse erzielt werden.

8.2 Diskussion, Einordnung und Ausblick

Das gesteckte Ziel der Entwicklung eines Lernverfahrens zur Lösung des vollständigen visuomotorischen Lernproblems wurde erreicht. Es wurde mit DFQ-C erfolgreich ein wertfunktionsbasiertes Reinforcement Lernverfahren direkt auf unvorverarbeitete Bilder angewendet, um ohne weiteres Vorwissen Strategien zur Steuerung realer Systeme zu erlernen. Es wurde nicht nur eine Simulation mit verrauschten, synthetischen Bildern verwendet, sondern auch eine reale Bildformation in einer videospielähnlichen, im Rechner simulierten Problemstellung.

Stabilität, Effizienz, Generalisierung Unter den drei bereits in der Einleitung formulierten und in den Kapiteln 4 und 5 verfeinerten Bewertungskriterien – Stabilität, Effizienz und Generalisierung – kann DFQ-C überzeugen. Das stabile Verhalten konnte aufgrund des Einsatzes des konstanten irregulären Gitterapproximators trotz der Verwendung nicht linearer, tiefer Encodernetze für diskontierte Problemstellungen formal nachgewiesen und in allen Lernexperimenten praktisch beobachtet werden. Auch wenn die stochastische Konsistenz nicht formal bewiesen werden kann, sind die praktischen Voraussetzungen für wiederholbare, stabile und sich kontinuierlich verbessernde Lernverläufe mit wenigen Schwankungen offensichtlich gegeben. Aufgrund der Verbindung

mit den batch RL-Verfahren ist DFQ sehr dateneffizient und es entsteht nur ein sehr geringer Mehraufwand gegenüber den klassischen Verfahren durch die Verwendung der tiefen Autoencoder und die gelegentlichen Generationswechsel. Nach einer anfänglichen Explorationsphase ist die erlernte Repräsentation oftmals so gut, dass das vom batch RL zu lösende Strategielernproblem kaum schwieriger ist, als beim Einsatz einer klassischen Bildverarbeitung. Die von den tiefen Encodern erzeugten Merkmalsräume wurden in Kapitel 5 ausführlich hinsichtlich der sich bietenden Generalisierungsmöglichkeiten untersucht. In den untersuchten Navigationsexperimenten wurden Merkmalsvektoren ähnlicher Observationen auf benachbarte Positionen im Merkmalsraum abgebildet, sie waren klar von anderen Zuständen abzugrenzen und robust gegenüber Rauschen. Selbst die Topologie wurde teilweise gut in den Merkmalsraum übertragen. Abgesehen von der unterschiedlichen Anordnung sind die Merkmalsvektoren nicht prinzipiell schlechter als die nicht beobachtbaren Zustände für eine Generalisierung geeignet. Mit den irregulären Gittern in ClusterRL wurde ein Verfahren entwickelt, das aus den vorab unbekannten Anordnungen im Merkmalsraum über eine automatische Anpassung an die Lage der Daten und lokale Dichten einen Nutzen ziehen kann und zu guten Generalisierungsergebnissen führt.

Bezugspunkt klassischer Ansatz Die durch parallele Verarbeitung beschleunigte Implementierung der tiefen Netze und DFQs erlaubt die Anwendung von DFQ-C auf reale, dynamische Systeme mit harten Echtzeitanforderungen. Das durchgeführte Experiment auf der Carrerabahn ist die erste, erfolgreiche Anwendung Reinforcement Lernens direkt auf unverarbeitete Bilder zur Steuerung eines realen, dynamischen Systems. Lerndauer und Qualität der erlernten Strategien sind verlgeichbar zu den mittels des klassischen Ansatzes mit handkodierter Bildverarbeitung erzielten Ergebnissen [74] – was eine wichtige Forderung aus Abschnitt 1.5 war.

Verwandte Arbeiten Im Vergleich zu den eingangs in Abschnitt 1.4 diskutierten Lernverfahren stellt DFQ-C einen deutlichen Schritt nach vorne dar (siehe Tabelle 8.1). Es erlaubt die "lokale" Generalisierung auf zuvor nicht gesehene, ähnliche Bilder, hat sich als robust gegen Bildrauschen erwiesen und bietet in der Praxis – für die inneren Schleife auch nachweislich – stabile Lernvorgänge. Es ist zudem das einzige Verfahren, das direkt auf den Bildern arbeitet, die Merkmalsextraktion (V-x & V-s) vollständig erlernt und bereits auch erfolgreich auf reale Systeme angewendet wurde.

Das Kritierium "Informationserhaltung" bei der Durchführung der Daten durch einen engen Flaschenhals hat sich in dieser Arbeit allgemein als sehr leistungsfähig beim selbständigen Erlernen von informationstragenden, kompakten Repräsentationen erwiesen. Dieses Kriterium fällt in eine Kategorie von anderen, derzeit im Bereich der Neurowissenschaften untersuchten Optimierungskriterien zur selbstorganisierten Herausbildung semantisch gehaltvoller Repräsentationen. Zu nennen sind hier die Kriterien "Slowness" bzw. "Temporal Stability" [9, 174] und "Sparsity" [109], mit denen zum Beispiel Repräsentationen unüberwacht erlernt wurden, die den bei Säugetieren gefundenen Ortsneuronen ähneln [175]. Der primäre Vorteil der hier eingesetzten Autoen-

coder besteht zum einen in der ausschließlich lokalen Optimierung – die Berechnung von Aktivierungen und Anpassungen einzelner Neuronen müssen nicht die Ausgaben und Fehler aller Nachbarneuronen in derselben Schicht berücksichtigen – und zum anderen im Rückgriff auf ausschließlich – auch formal – gut untersuchte und verstandene Lernverfahren.

Tabelle 8.1: Vergleich verschiedener Ansätze unter den eingangs formulierten Bewertungskriterien zur Generalisierung (Spalten "Generalisierung", mit drei Abstufungen $+/\circ/-$), den durch Lernen gelösten Teilproblemen (V-x: Merkmalsextraktion, V-s: Merkmalsselektion, M: Modell, S: Strategie, $\bullet/\circ$ für vorhanden / nicht vorhanden), der Stabilität des Lervorgangs (drei Abstufungen $+/\circ/-$) und der Erprobung auf realen Systemen ($\bullet/\circ$ für erfolgreich durchgeführt / nicht durchgeführt).

	Teilprobleme				Stabiles	Generalisierung			
Ansatz	V-x	V-s	M	S	Lernen	Robust	Lokal	Topo	Real
Klassisch	○		●	●	+	+	+	+	●
ALVINN	●		○	○[1]	+	○	○	?	●
evoVision	●[2]	●	○	○[1]	○	+	+	?	●
Gordon	●		○	●	+	+	−	−	○
Ernst	●		●	●	+	○	−	−	○
Dozono	●		●	●	−[3]	+	○	○	○
Jodonge	○	●	●	●	+	+[4]	○	−	○
DFQ	●		●	●	○ / +[5]	+	+	○ /+[6]	●

DFQ in praktischen Anwendungen Eine Beschränkung des erarbeiteten Verfahrens besteht derzeit noch in der verarbeitbaren Bildgröße. Im Rahmen dieser Arbeit wurden bereits 1/4-QVGA Bilder von der Carrerabahn mit bis zu 80×60 Pixeln verarbeitet, was gegenüber den Bildgrößen des MNIST-Datensatzes mit 28×28 Pixeln einer fünffachen Steigerung entspricht. Auch das Training von Netzen auf noch größeren Bildern ist mit Faltungskernen durchaus möglich. In einem vorläufigen Experiment mit einer längeren Carrerabahn und lediglich 2000 Trainingsbildern mit $140 \times 120 = 16\,800$ Pixeln wurden gute Ergebnisse erzielt (siehe Abbildung 8.1). In der aktuelleren Literatur sind Bildgrößen bis zu 8976 Pixeln in Objektklassifikationsaufgaben zu finden [106]. Netze dieser Größenordnung stoßen aber definitiv an die Grenzen des auf aktuellen Workstations derzeit Machbaren. Aufgrund der zuletzt immer schneller entwickelten parallelen Rechnerarchitekturen, insbesondere auf Grafikkarten [123, 153], verschieben

[1] Überwachtes Lernen

[2] Lernt aufgabenspezifische Parameter vorgegebener Algorithmen

[3] Maximiert sofortige Belohnungen, starkes Reward-Shaping

[4] Allerdings Eigenschaft des Merkmalsextraktors, nicht des Lernvorgangs

[5] Schwankungen durch Generationswechsel. Bei rauschfreier (künstl.) Bildformation genauso stabil wie die Verfahren von Jodogne, Ernst und Gordon

[6] Methoden zur Verbesserung vorgestellt

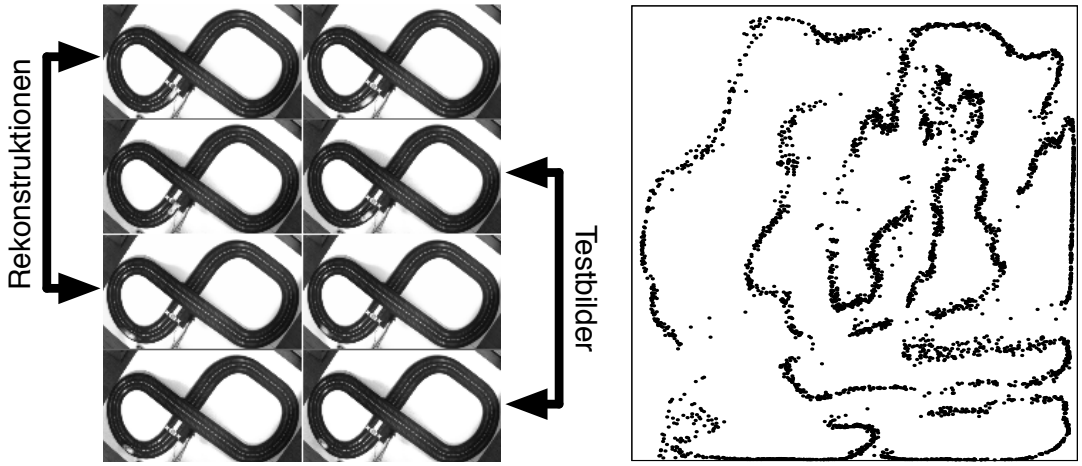


Abbildung 8.1: Verarbeitung großer Bilder mit 140×120 Pixeln einer längeren Carrerabahn. Links: Rekonstruktionen von Testbildern. Rechts: Verteilung der Merkmalsvektoren.

sich die Grenzen aber stetig. Mit den tiefen Netzen bietet sich ein lohnenswertes Einsatzgebiet für diese neuen, parallelen Rechnerarchitekturen.

Ein praktische Anwendungen erschwerendes Problem ist – wie bei vielen klassischen oder lernenden Bilderkennungs- und Bildklassifikationsverfahren auch – die Anfälligkeit gegenüber Beleuchtungsänderungen. Allerdings verhält sich die Repräsentation bei Beleuchtungsänderungen anders als man vermuten könnte: Veränderte Helligkeiten und Kontraste führten bei der Carrerabahn nicht zu einem völligen Ausreißen der Aktivierungen und etwa chaotischen Änderungen im Merkmalsraum, sondern verschoben die Merkmalsvektoren nur leicht, ohne die grundsätzliche Anordnung in einem Band zu zerstören (siehe Abbildung 8.2). Dieses interessante Phänomen wurde in zwei unabhängigen Wiederholungen beobachtet und könnte in Zukunft für eine Adaptation an Beleuchtungsänderungen ausgenutzt werden.

In der vorliegenden Arbeit wurden die tiefen Autoencodernetze und DFQ bisher ausschließlich auf RL-typischen Navigationsproblemen getestet, in denen sich nur ein zu steuerndes Objekt bewegt. Dadurch wird zwar auch eine große Klasse regelungstechnischer Systeme und typischer Benchmarkaufgaben wie das Mountain-Car, das unteraktuierten und das inversen Pendel erfasst, aber viele interessante Systeme bleiben außen vor. In Zukunft ist hier durchaus eine Erweiterung auf komplexere Situationen denkbar. Der nächste Schritt wäre zum Beispiel das Lösen von Aufgaben, bei denen ein Agent ein sich bewegendes Ziel erreichen oder verfolgen muss. Um solchen Aufgabenstellungen mehrere Objekte verfolgen und im Merkmalsraum repräsentieren zu können, müssten wie in [88, 124] komplexere Netze mit mehreren unabhängigen Faltungskernen in den äußeren Schichten eingesetzt werden. Erste Ergebnisse eines solchen Autoencoders mit vier Neuronen in der Kodierungsschicht zur Kodierung der Position von zwei unterschiedlichen Objekten sind in Abbildung 8.3 zu sehen.

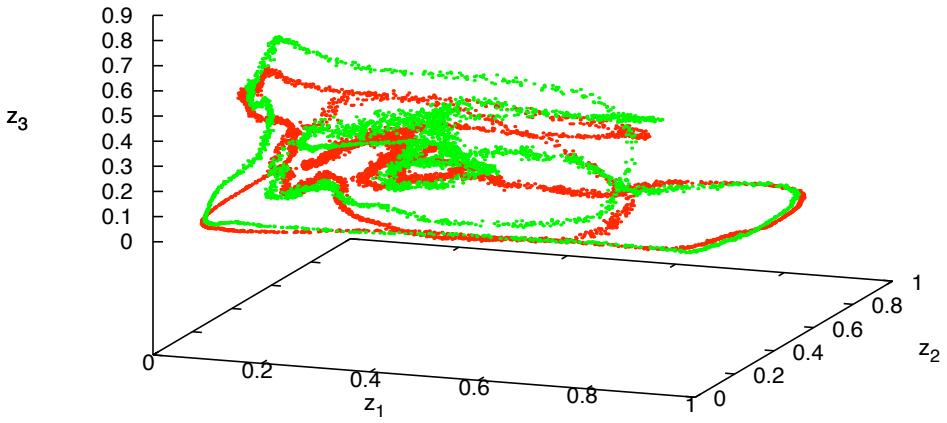


Abbildung 8.2: Veränderung der Lage der Merkmalsvektoren bei Beleuchtungsänderungen. Mermkalsvektoren vor (rot) und nach (grün) der Änderung.

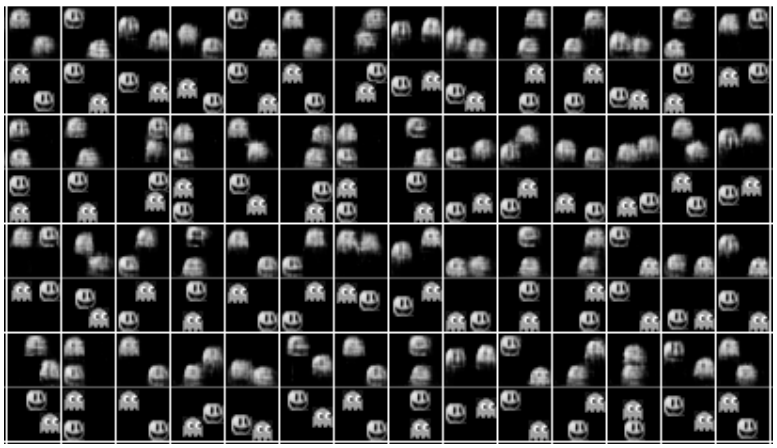


Abbildung 8.3: Testbilder (in den geraden Zeilen) mit zugehöriger Rekonstruktion jeweils direkt darüber (in den ungeraden Zeilen). Die vom Autoencoder mit vier Neuronen in der Kodierungsschicht erzeugten Rekonstruktionen sind noch nicht perfekt, die Figuren aber immer an der richtigen Position und meistens gut zu unterscheiden.

Invarianzen und komplexe visuelle Verarbeitung Generell ist durch die Verwendung der Faltungskerne das Erlernen von posititions- bzw. translationsinvarianten Merkmalsdetektoren möglich, bzw. wurde bereits im Rahmen dieser Arbeit erzielt, aber Rotationsinvarianzen sind mit den üblichen Netzarchitekturen allein schwer zu erzielen und ein noch offenes Problem. Interessant ist auch die Frage, ob sich Autoencodernetze für die Verarbeitung von Bildern einer sich im Raum bewegenden, auf dem Agenten montierten Kamera eignen – kann hier Information effektiv komprimiert werden? Längerfristig wären wesentlich komplexere Netze denkbar, die eine solche Verarbeitung

ermöglichen. Dabei könnte man sich weiter an der Natur orientieren und Verbindungsstrukturen in der Retina des menschlichen Auges nachempfinden. Statt den tiefen Lernverfahren alle Freiheitsgrade zu überlassen, könnten auch die Verknüpfungen der Zapfen über die Ganglien bis in die rezeptiven Felder des Corpus geniculatum laterale und die Orientierungskolumnen des primären visuellen Cortex [72] nachempfunden werden. Dazu könnten Gewichte in den äußeren Schichten fest eingestellt oder zum Beispiel auf das Ausformen von on/off-Center Zellen vortrainiert werden. Nur die höheren Netzschichten würden dann auf Basis dieser biologisch motivierten, ikonischen Repräsentation aufgabenspezifisch trainiert. Ein Problem bei solchermaßen großen Netzen sind momentan aber noch neben der Laufzeit (s.o.) die Auswahl passender Modelle und die Feineinstellung der Parameter – beides ist noch mehr eine Kunst denn eine Wissenschaft. Die Untersuchungen [38, 83] sind erste Schritte in Richtung eines besseren Verständnisses und der Entwicklung eines systematischeren Vorgehens, viele weitere Schritte müssen folgen.

Wertfunktionsbasiertes Reinforcement Lernen Hinsichtlich des Reinforcement Lernens ergeben sich durch diese Arbeit neue, interessante Einsatzgebiete für batch RL Verfahren. Der Weg, einen für direktes Lernen zu großen Zustandsraum automatisch auf einen Merkmalsraum mit weniger Dimensionen zu komprimieren, wo mit FQI, LSPI und NFQ leistungsfähige Verfahren zur Verfügung stehen, erscheint auch ausserhalb des visuellen Lernens attraktiv. Es gibt bereits erste Versuche, mehrschichtige, abstrahierende Modelle von menschlichen Bewegungen mittels RBMs zu erlernen [158, 159, Abschnitt 5]. Diese Technik könnte auch für RL-Anwendungen wie die Steuerung eines Roboterarms – derzeit den Policy Search und Policy Gradient Verfahren vorbehalten [115] – interessant sein.

Das auf den in DFQ eingesetzten irregulären Gitterapproximatoren basierende ClusterRL-Verfahren ist ein allgemein interessantes batch Verfahren, das sich weitestgehend automatisch an die Lage der Daten im Zustandsraum anpassen kann. Insbesondere in Hinblick auf DFQ ist diese Eigenschaft ein wichtiger Beitrag zur Kompatibilitätserhöhung [51, 53] der Gitterapproximatoren, die Voraussetzung für die Konvergenz in nicht diskontierten Verfahren und allgemein für gute Approximationsergebnisse ist. Die prinzipielle Motivation kam aus der iterativen Aufsplittung der regulären Gitter nach [104, 126], bzw. aus deren Unzulänglichkeiten. Ziel war die Entwicklung eines einfacheren, rein datengetriebenen Verfahrens. Tatsächlich muss hier aber kein Gegensatz zwischen diesen Verfahren gesehen werden. Verfahren der Clusteranalyse könnten zum schnellen Finden guter initialer irregulärer Gitter dienen, die dann mit Hilfe der Splittingkriterien von Munos [104] und insbesondere Reynolds [126] weiter verfeinert werden könnten. Diese Kriterien sind nicht auf den Einsatz in zunächst regulären, als kd-Bäume realisierbaren Gitterstrukturen beschränkt. Sie können genauso in irregulären Gittern zum Einsatz kommen und aufgrund der einfacherern Aufteilungsoperationen sogar leichter umgesetzt werden. Batch RL bietet zudem die Möglichkeit, Daten direkt zu analysieren und Varianzen und andere Kriterien direkt zu berechnen, statt sie online schätzen zu müssen [126].

Im Zusammenhang mit den speicherbasierten batch RL-Verfahren bietet sich eine Vielzahl von Möglichkeiten, von der Speicherung der Daten zu profitieren, die in reinen online Verfahren nicht bestehen. Mit dem Training der Autoencoder zum Erlernen von für die Daten geeigneten Einbettungen in niedrigdimensionale Merkmalsräume und der automatischen Anpassung von irregulären Gitterstrukturen durch Verfahren der Clusteranalyse wurde von diesen neuen Möglichkeiten auch extensiv Gebrauch gemacht. Ein in dieser Arbeit mit der bipolaren Explorationsstrategie (siehe Abschnitt 7.1.3) nur am Rande untersuchtes Thema ist die zielgerichtete Steuerung der Exploration. Batch RL-Verfahren bieten hier die Möglichkeit zur globalen zählerbasierten Exploration [162, 163]. Den ersten Eindrücken nach (siehe auch Abbildung 8.4) erscheint auch diese Untersuchungsrichtung als vielversprechend.

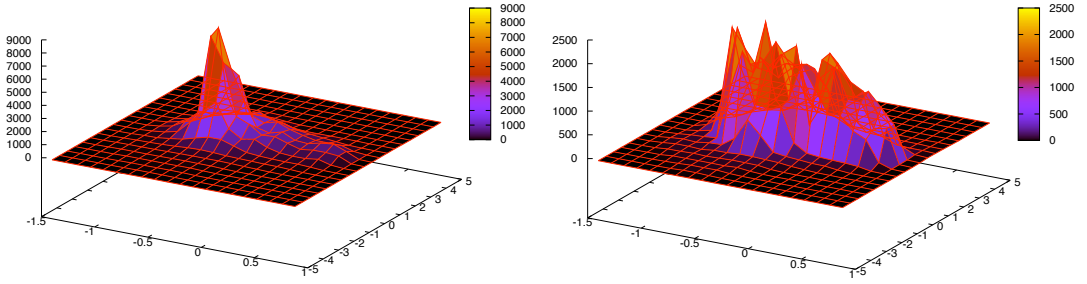


Abbildung 8.4: 2D-Histogramm über die Verteilung der besuchten Zustände unter Anwendung verschiedener Explorationsstrategien im Mountain-Car-Problem (1000 Episoden à maximal 100 Schritte). Links: Klassische Vorgehensweise mit optimistischer Initialisierung und ϵ-greedy Exploration ($\epsilon = 0.3$). Rechts: Zielgerichtete, globale Exploration, zählerbasiert, Suchstrategie mittels batch RL programmiert. Achtung: Skalenwechsel!

8.3 Fazit

Die Relevanz tiefer Autoencoder für das optimierende Lernen konnte in dieser Arbeit klar bestätigt werden. DFQ ist ein erster Algorithmus, der von der Leistungsfähigkeit der Autoencoder profitiert, das Lernen auf hochdimensionalen Eingabedaten ermöglicht und so die Grenzen für das wertfunktionsbasierte Reinforcement Lernen deutlich verschiebt. DFQ wurde nicht nur erfolgreich auf realistische, aber synthetische Bilddaten angewendet, sondern es wurden bereits auch eindrucksvolle Ergebnisse auf realen Anwendungen erzielt, die sich mit den Ergebnissen klassischer Ansätze messen lassen können. DFQ hat sich hierbei als vielversprechender Ansatz erwiesen mit vielen sich eröffnenden, weiterführenden Forschungsmöglichkeiten. Die eigenen Erwartungen wurden aber bereits jetzt deutlich übertroffen. Es ist nun möglich, direkt auf unvorverarbeiteten Bilddaten optimierendes Lernen zu betreiben und so gute Strategien zur Steuerung realer Systeme direkt auf Basis visueller Wahrnehmungen zu erlernen.

Anhang A

Beweis der Eindeutigkeit des abgeleiteten Zufallsoperators für Observationen

Beweis der Eindeutigkeit: Nehmen wir an, es gäbe zwei unterschiedliche Zufallsoperatoren $\widetilde{H}^a_{dp}$ und $\widetilde{K}^a_{dp}$ mit unterschiedlichen Kerneln $k'(o,o')$ und $k''(o,o)$ – es existiere also mindestens ein Paar $(o_1,o_2) \in O \times O$ mit $k'(o_1,o_2) \neq k''(o_1,o_2')$ – die beide zu H^a_{dp} korrespondieren, also gelte $\widetilde{H}^a_{dp}V'(o) = \hat{H}^a_{dp}V(z)$ respektive $\widetilde{K}^a_{dp}V'(o) = \hat{H}^a_{dp}V(z)$ für alle $o \in O$ und $z = \mathrm{ENC}(o;W)$. Sei nun $\mathcal{F}_{O,a}$ eine Menge von Übergängen, die auch einen Übergang (o_1,a_1,r_1,o_1') enthält. Tauche o.B.d.A. o_1' zudem in keinem weiteren Übergang in $\mathcal{F}_{O,a}$ als Zielzustand auf. Aus Gleichsetzen und einfachen Äquivalenzumformungen folgt für die Stelle o_2:

$$\widetilde{H}^a_{dp}V'(o_0) = \widetilde{K}^a_{dp}V'(o_0) \tag{A.1}$$

$$\Leftrightarrow \widetilde{H}^a_{dp}V'(o_0) - \widetilde{K}^a_{dp}V'(o_0) = 0 \tag{A.2}$$

$$\Leftrightarrow \sum_{(o,a,r,o')\in\mathcal{F}_{O,a}} k'(o,o_0)[r+\gamma V'(o')] - \sum_{(o,a,r,o')\in\mathcal{F}_{O,a}} k''(o,o_0)[r+\gamma V'(o')] = 0 \tag{A.3}$$

$$\Leftrightarrow \sum_{(o,a,r,o')\in\mathcal{F}_{O,a}} \big(k'(o,o_0)[r+\gamma V'(o')] - k''(o,o_0)[r+\gamma V'(o')]\big) = 0 \tag{A.4}$$

$$\Leftrightarrow \sum_{(o,a,r,o')\in\mathcal{F}_{O,a}} [r+\gamma V'(o')]\,\big(k'(o,o_0) - k''(o,o_0)\big) = 0\ . \tag{A.5}$$

Es lässt sich nun der Übergang (o_1,a_1,r_1,o_1') aus der Summe herausziehen und es folgen die zu (A.5) äquivalenten Gleichungen

$$[r_1+\gamma V'(o_1')]\,\big(k'(o_1,o_2) - k''(o_1,o_2)\big) + \sum_{(o,a,r,o')\in\mathcal{F}_{O,a}\setminus(o_1,a_1,r_1,o_1')} [r+\gamma V'(o')]\,\big(k'(o,o_0) - k''(o,o_0)\big) = 0$$

$$\Leftrightarrow [r_1 + \gamma V'(o_1')] \left(k'(o_1, o_2) - k''(o_1, o_2)\right) = -\sum_{(o,a,r,o') \in \mathcal{F}_{O,a} \setminus (o_1, a_1, r_1, o_1')} [r + \gamma V'(o')] \left(k'(o, o_0) - k''(o, o_0)\right) \ . \tag{A.6}$$

Führt man nun die Schritte (A.1–A.6) für eine zweite Wertfunktion $V''(o)$ durch, die sich bloss an der Stelle o_1' von $V'(o)$ unterscheidet – konkret $V''(o_1') = V'(o_1') + 1$ und ansonsten identisch ist, erhält man

$$\begin{aligned} [r_1 + \gamma V''(o_1')] \left(k'(o_1, o_2) - k''(o_1, o_2)\right) &= -\sum_{(o,a,r,o') \in \mathcal{F}_{o,a} \setminus (o_1, a_1, r_1, o_1')} [r + \gamma V''(o')] \left(k'(o, o_0) - k''(o, o_0)\right) \\ [r_1 + \gamma(V'(o_1') + 1)] \left(k'(o_1, o_2) - k''(o_1, o_2)\right) &= -\sum_{(o,a,r,o') \in \mathcal{F}_{o,a} \setminus (o_1, a_1, r_1, o_1')} [r + \gamma V'(o')] \left(k'(o, o_0) - k''(o, o_0)\right) \ . \end{aligned} \tag{A.7}$$

Aufgrund (A.6) und (A.7) müsste also gelten:

$$\begin{aligned} [r_1 + \gamma(V'(o_1') + 1)] \left(k'(o_1, o_2) - k''(o_1, o_2)\right) &= [r_1 + \gamma V'(o_1')] \left(k'(o_1, o_2) - k''(o_1, o_2)\right) \\ [\gamma V'(o_1') + \gamma] \left(k'(o_1, o_2) - k''(o_1, o_2)\right) &= [\gamma V'(o_1')] \left(k'(o_1, o_2) - k''(o_1, o_2)\right) \\ \gamma \left(k'(o_1, o_2) - k''(o_1, o_2)\right) &= 0 \\ k'(o_1, o_2) &= k''(o_1, o_2) \ , \end{aligned}$$

was im offensichtlichen Widerspruch zu $k'(o_1, o_2) \neq k''(o_1, o_2')$ steht. D.h. der in Lemma 5 konstruierte Kernel ist die einzige allgemeingültige Lösung von $\widetilde{H}_{dp}^{a} V'(o_\rangle) = \hat{H}_{dp}^{a} V(z)$ für alle $o \in O$ und zugehörige $z = \text{ENC}(o; W)$, die für alle Wertfunktionen $V : 0 \mapsto \mathbb{R}$ gilt. □

Literatur

[1] Ackley, D.H. ; Hinton, G.E. ; Sejnowski, T.J.: A learning algorithm for Boltzmann machines. In: *Cognitive science* 9 (1985), Nr. 1, S. 147–169

[2] Anderson, CW: Learning to control an inverted pendulum using neural networks. In: *IEEE Control Systems Magazine* 9 (1989), Nr. 3, S. 31–37

[3] Asada, M. ; Noda, S. ; Tawaratsumida, S. ; Hosoda, K.: Purposive behavior acquisition for a real robot by vision-based reinforcement learning. In: *Machine Learning* 23 (1996), Nr. 2, S. 279–303

[4] Baird, L.: Residual algorithms: Reinforcement learning with function approximation. In: *Proc. of the 12th International Conference on Machine Learning*, 1995, S. 30–37

[5] Bakker, B. ; Zhumatiy, V. ; Gruener, G. ; Schmidhuber, J.: Quasi-online reinforcement learning for robots. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006, S. 2997–3002

[6] Baldi, P. ; Homik, K.: Neural networks and principal component analysis: learning from examples without local minima. In: *Neural Networks* 2 (1989), Nr. 1, S. 53–58

[7] Balkenius, C. ; Johansson, B.: Anticipatory models in gaze control: a developmental model. In: *Cognitive Processing* 8 (2007), Nr. 3, S. 167–174

[8] Bathe, K.J.: *Finite-Elemente-Methoden*. Springer, 2002

[9] Becker, S.: Implicit learning in 3D object recognition: The importance of temporal context. In: *Neural Computation* 11 (1999), Nr. 2, S. 347–374

[10] Bellman, R. E.: *Dynamic Programming*. Princeton University Press, 1957

[11] Bengio, Y. ; Lamblin, P. ; Popovici, D. ; Larochelle, H. ; Montreal, Q.: Greedy Layer-Wise Training of Deep Networks. In: *Advances in Neural Information Processing Systems* Bd. 19, 2007

[12] Bengio, Yoshua: Learning deep architectures for AI / Dept. IRO, Universite de Montreal, TR 1312. 2007. – Forschungsbericht

[13] Berridge, S. ; Schumacher, J. M.: An irregular grid method for high-dimensional free-boundary problems in finance. In: *Future Generation Computer Systems* 20 (2004), Nr. 3, S. 353–362

[14] Bertsekas, D.P.: Dynamic programming and optimal control. (1995)

[15] Bertsekas, D.P. ; Tsitsiklis, J.N.: An analysis of stochastic shortest path problems. In: *Mathematics of Operations Research* 16 (1991), Nr. 3, S. 580–595

[16] Bertsekas, D.P. ; Tsitsiklis, J.N.: Neuro-dynamic programming. (1996)

[17] Bishop, C.M.: Neural Networks for Pattern Recognition. (1995)

[18] Blackford, L.S. ; Demmel, J. ; Dongarra, J. ; Duff, I. ; Hammarling, S. ; Henry, G. ; Heroux, M. ; Kaufman, L. ; Lumsdaine, A. ; Petitet, A. u. a.: An updated set of basic linear algebra subprograms (BLAS). In: *ACM Transactions on Mathematical Software* 28 (2002), Nr. 2, S. 135–151

[19] Bourlard, H. ; Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. In: *Biological cybernetics* 59 (1988), Nr. 4, S. 291–294

[20] Boyan, J. ; Moore, A.W.: Generalization in reinforcement learning: Safely approximating the value function. In: *Advances*

in neural information processing systems (1995), S. 369–376

[21] Bryant, R.E.: Symbolic Boolean manipulation with ordered binary-decision diagrams. In: *ACM Computing Surveys (CSUR)* 24 (1992), Nr. 3, S. 293–318

[22] Caruana, Richard A.: Multitask learning: A knowledge-based source of inductive bias. In: *Proc. of the 10th International Conference on Machine Learning*, 1993, S. 41–48

[23] Chapman, B. ; Godecke, I. ; Bonhoeffer, T.: Development of orientation preference in the mammalian visual cortex. In: *Journal of Neurobiology* 41 (1999), Nr. 1, S. 18–24

[24] Chopra, S. ; Hadsell, R. ; LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: *IEEE Computer Vision and Pattern Recognition (CVPR)* Bd. 1, 2005, S. 539–546

[25] Chow, CS ; Tsitsiklis, JN: An optimal multigrid algorithm for discrete-time stochastic control / Technical Report P-135, Center for Intelligent Control Systems, 1989. – Forschungsbericht

[26] Chow, C.S. ; Tsitsiklis, J.N.: An optimal one-way multigrid algorithm for discrete-time stochastic control. In: *IEEE Transactions on Automatic Control* 36 (1991), Nr. 8, S. 898–914

[27] Cottrell, GW ; Munro, PW ; Zipser, D.: Image compression by back propagation: A demonstration of extensional programming. In: *Advances in cognitive science* 2 (1987)

[28] Cottrell, M. ; Hammer, B. ; Hasenfuss, A. ; Villmann, T.: Batch and median neural gas. In: *Neural Networks* 19 (2006), Nr. 6-7, S. 762–771

[29] Crites, R. ; Barto, A.: Improving Elevator Performance using Reinforcement Learning. In: *Advances in Neural Information Processing Systems 8*, 1995, S. 1017–1023

[30] Delaunay, B.: Sur la sphère vide. In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7 (1934), S. 793–800

[31] Diamond, A.: Close Interrelation of Motor Development and Cognitive Development and of the Cerebellum and Prefrontal Cortex. In: *Child Development* 71 (2000), Nr. 1, S. 44–56

[32] Doischer, D. ; Aurel Hosp, J. ; Yanagawa, Y. ; Obata, K. ; Jonas, P. ; Vida, I. ; Bartos, M.: Postnatal Differentiation of Basket Cells from Slow to Fast Signaling Devices. In: *The Journal of Neuroscience* 28 (2008), Nr. 48, S. 12956–12968

[33] Dongarra, J.J. ; Du Croz, J. ; Hammarling, S. ; Duff, IS: A set of level 3 basic linear algebra subprograms. In: *ACM Transactions on Mathematical Software (TOMS)* 16 (1990), Nr. 1, S. 1–17

[34] Dongarra, J.J. ; Du Croz, J. ; Hammarling, S. ; Hanson, R.J.: An extended set of FORTRAN basic linear algebra subprograms. In: *ACM Transactions on Mathematical Software* 14 (1988), Nr. 1, S. 1–17

[35] Dozono, H. ; Fujiwara, R. ; Takahashi, T.: Application of the self organizing maps for visual reinforcement learning of mobile robot. In: *Proc. of the 7th WSEAS International Conference on Artificial intelligence, knowledge engineering and data bases (AIKED'08)*. Stevens Point, Wisconsin, USA, 2008, S. 257–262

[36] Dozono, H. ; Fujiwara, R. ; Takahashi, T.: Visual reinforcement learning algorithm using self organizing maps and its simulation in OpenGL environment. In: *WSEAS Transactions on Information Science and Applications* 5 (2008), Nr. 5, S. 685–694

[37] Elman, J.L. ; Zipser, D.: Learning the hidden structure of speech. In: *The Journal of the Acoustical Society of America* 83 (1988), S. 1615

[38] Erhan, D. ; Manzagol, P.A. ; Bengio, Y. ; Bengio, S. ; Vincent, P.: The difficulty of training deep architectures and the effect of unsupervised pre-training. In: *Proc.*

of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS), 2009

[39] Ernst, D. ; Geurts, P. ; Wehenkel, L.: Tree-Based Batch Mode Reinforcement Learning. In: *Journal of Machine Learning Research* 6 (2005), Nr. 1, S. 503–556

[40] Ernst, D. ; Marée, R. ; Wehenkel, L.: Reinforcement learning with raw pixels as input states. In: *International Workshop on Intelligent Computing in Pattern Analysis/Synthesis (IWICPAS)*, 2006, S. 446–454

[41] Fernandez, F. ; Borrajo, D.: VQQL. Applying vector quantization to reinforcement learning. In: *Robocup-99: Robot Soccer World Cup III*, 2000, S. 292–303

[42] Forsyth, D.A. ; Ponce, J.: *Computer vision: A modern approach.* Prentice Hall, 2002

[43] Freund, Y. ; Haussler, D.: Unsupervised learning of distributions on binary vectors using two layer networks. In: *Advances in Neural Information Processing Systems 4*, 1991

[44] Fritzke, B.: A growing neural gas network learns topologies. In: *Advances in Neural Information Processing Systems 7* (1995)

[45] Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. In: *Biological Cybernetics* 36 (1980), Nr. 4, S. 193–202

[46] Gabel, T. ; Hafner, R. ; Lange, S. ; Lauer, M. ; Riedmiller, M.: Bridging the gap: Learning in the RoboCup simulation and midsize league. In: *Proc. of the 7th Portuguese Conference on Automatic Control*, 2006

[47] Gabel, T. ; Riedmiller, M.: Adaptive Reactive Job-Shop Scheduling with Learning Agents. In: *International Journal of Information Technology and Intelligent Computing* 2 (2007), Nr. 4

[48] Gaskett, C. ; Fletcher, L. ; Zelinsky, A.: Reinforcement learning for a vision based mobile robot. In: *Proc. of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000

[49] Gaskett, C. ; Fletcher, L. ; Zelinsky, A.: Reinforcement learning for visual servoing of a mobile robot. In: *Proc. of the Australian Conference on Robotics and Automation (ACRA)*, 2000

[50] Goldsmith, T.H.: Optimization, Constraint, and History in the Evolution of Eyes. In: *The Quarterly Review of Biology* 65 (1990), Nr. 3, S. 281

[51] Gordon, G.: Stable Function Approximation in Dynamic Programming. In: *Proc. of the Twelfth International Conference on Machine Learning*. Tahoe City, USA, 1995, S. 261–268

[52] Gordon, G.J.: Stable fitted reinforcement learning. In: *Advances in Neural Information Processing Systems* (1995), S. 1052–1058

[53] Gordon, G.J.: Stable function approximation in dynamic programming. CMU School of Computer Science, Pittsburgh, PA, Tech Report CMU-CS-95-103, 1995. – Forschungsbericht

[54] Gordon, G.J.: Chattering in SARSA (λ). CMU Learning Lab, Pittsburgh, PA, Internal Report, 1996. – Forschungsbericht

[55] Gouet, V. ; Boujemaa, N.: Object-based queries using color points of interest. In: *IEEE Workshop on Content-Based Access of Image and Video Libraries*, 2001, S. 30–36

[56] Gross, H.M. ; Stephan, V. ; Krabbes, M.: A neural field approach to topological reinforcement learning incontinuous action spaces. In: *Proc. of the International Joint Conference on Neural Networks*, 1998

[57] Hammer, B. ; Villmann, T.: Effizient Klassifizieren und Clustern: Lernparadigmen von Vektorquantisierern. In: *Künstliche Intelligenz* 6 (2006), Nr. 3, S. 5–11

[58] Harris, C. ; Stephens, M.: A combined corner and edge detector. In: *Fourth Alvey Vision Conference* Manchester, UK, 1988, S. 147–151

[59] Hinton, G.E.: Training products of experts by minimizing contrastive divergence. In: *Neural Computation* 14 (2002), Nr. 8, S. 1771–1800

[60] Hinton, G.E. ; Osindero, S. ; Teh, Y.W.: A Fast Learning Algorithm for Deep Belief Nets. In: *Neural Computation* 18 (2006), Nr. 7, S. 1527–1554

[61] Hinton, G.E. ; Salakhutdinov, R.R.: Reducing the Dimensionality of Data with Neural Networks. In: *Science* 313 (2006), Nr. 5786, S. 504–507

[62] Hubel, David H.: *Eye, brain, and vision.* Scientific American Library, 1988

[63] Igel, C. ; Husken, M.: Empirical evaluation of the improved Rprop learning algorithms. In: *Neurocomputing* 50 (2003), Nr. 1, S. 105–124

[64] Jodogne, S. ; Briquet, C. ; Piater, J.H.: Approximate Policy Iteration for Closed-Loop Learning of Visual Tasks. In: *Proc. of the European Conference on Machine Learning*, 2006

[65] Jodogne, S. ; Piater, J.H.: Interactive selection of visual features through reinforcement learning. In: *Proc. of the 24th SGAI Internal Conference on Innovative Techniques and Applications of Artificial Intelligence*, S. 285–298

[66] Jodogne, S. ; Piater, J.H.: Interactive learning of mappings from visual percepts to actions. In: *Proc. of the 22nd international conference on Machine learning*, 2005, S. 393–400

[67] Jodogne, S. ; Piater, J.H.: Learning, then Compacting Visual Policies. In: *7th European Workshop on Reinforcement Learning*, 2005, S. 8–10

[68] Jodogne, S. ; Piater, J.H.: Reinforcement learning of perceptual classes using Q learning updates. In: *Proc. of the 23rd International Multi-Conference on Artificial Intelligence and Applications*, 2005, S. 445–450

[69] Jodogne, S. ; Piater, J.H.: Closed-loop learning of visual control policies. In: *Journal of Artificial Intelligence Research* 28 (2007), S. 349–391

[70] Jolliffe, I. T.: *Principal Component Analysis.* Zweite Edition. Springer, 2002

[71] Jähne, B.: *Digitale Bildverarbeitung.* Springer, 2005

[72] Kandel, E.R. ; Schwartz, J.H. ; Jessell, T.M.: *Principles of Neural Science.* Vierte Edition. McGraw-Hill, 2000

[73] Katz, L. C. ; Shatz, C. J.: Synaptic Activity and the Construction of Cortical Circuits. In: *Science* 274 (1996), Nr. 5290, S. 1133–1138

[74] Kietzmann, T.C. ; Riedmiller, M.: The Neuro Slot Car Racer: Reinforcement Learning in a Real World Setting. In: *Proc. of the 8th International Conference on Machine Learning and Applications*, 2009

[75] Knuth, D.E.: *The art of computer programming. Vol. 3, Sorting and Searching.* Addison-Wesley Reading, MA, 1973

[76] Kohonen, T.: Self-organized formation of topologically correct feature maps. In: *Biological cybernetics* 43 (1982), Nr. 1, S. 59–69

[77] Kohonen, T.: *Springer Series in Information Sciences.* Bd. 30: *Self-Organizing Maps*. Zweite Edition. Springer, Heidelberg, 1997

[78] Kuhtz-Buschbeck, JP ; Stolze, H. ; Jöhnk, K. ; Boczek-Funcke, A. ; Illert, M.: Development of prehension movements in children: a kinematic study. In: *Experimental Brain Research* 122 (1998), Nr. 4, S. 424–432

[79] Lagoudakis, M. ; Parr, R.: Least-Squares Policy Iteration. In: *Journal of Machine Learning Research* 4 (2003), S. 1107–1149

[80] Lange, S.: *Verfolgung von farblich markierten Objekten in 2 Dimensionen.* Bachelor Thesis, Universität Osnabrück, 2001

[81] Lange, S. ; Riedmiller, M.: Evolution of Computer Vison Subsystems in Robot Navigation and Image Classification Tasks. In: *RoboCup-2004: Robot Soccer World Cup VIII*, Springer, 2004

[82] Lange, S. ; Riedmiller, M.: Appearance-Based Robot Discrimination Using Eigenimages. In: *RoboCup 2006: Robot Soccer World Cup X*, 2007

[83] Larochelle, H. ; Bengio, Y. ; Louradour, J. ; Lamblin, P.: Exploring strategies for training deep neural networks. In: *The Journal of Machine Learning Research* 10 (2009), S. 1–40

[84] Larochelle, H. ; Erhan, D. ; Courville, A. ; Bergstra, J. ; Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: *Proc. of the 24th International Conference on Machine Learning*, 2007, S. 473–480

[85] Lauer, M. ; Lange, S. ; Riedmiller, M.: Calculating the perfect match: an efficient and accurate approach for robot self-localization. In: *Lecture Notes in Computer Science* 4020 (2006), S. 142

[86] Lauer, M. ; Lange, S. ; Riedmiller, M.: Motion estimation of moving objects for autonomous mobile robots. In: *Kunstliche Intelligenz* 20 (2006), Nr. 1, S. 11–17

[87] Lawson, C.L. ; Hanson, R.J. ; Kincaid, D.R. ; Krogh, F.T.: Basic Linear Algebra Subprograms for Fortran Usage. In: *ACM Transactions on Mathematical Software* 5 (1979), Nr. 3, S. 308–323

[88] LeCun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P.: Gradient-based learning applied to document recognition. In: *Proc. of the IEEE* 86 (1998), Nr. 11, S. 2278–2324

[89] Lee, E.-J. ; Merwine, D.K. ; Mann, L.B. ; Grzywacz, N.M.: Ganglion cell densities in normal and dark-reared turtle retinas. In: *Brain Research* 1060 (2005), S. 40–46

[90] Lee, Y. A. ; Hong, S. M.: Fuzzy Q-Map Algorithm for Reinforcement Learning. In: *International Conference on Computational Intelligence and Security (CIS2006)*, 2007, S. 298–307

[91] Lin, L.: Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. In: *Machine Learning* 8 (1992), Nr. 3, S. 293–321

[92] Lloyd, S.P.: Least squares quantization in PCM. In: *IEEE Transactions on Information Theory* 28 (1982), Nr. 2, S. 129–137

[93] Lowe, D.G.: Object recognition from local scale-invariant features. In: *International Conference on Computer Vision* Bd. 2, 1999, S. 1150–1157

[94] Martinetz, T.M. ; Berkovich, S.G. ; Schulten, K.J.: 'Neural-Gas' Network for Vector Quantization and its Application to Time-Series Prediction. In: *IEEE transactions on Neural Networks* 4 (1993), Nr. 4, S. 558–569

[95] McCormick, S. ; Thomas, J.: The fast adaptive composite grid (FAC) method for elliptic equations. In: *Mathematics of Computation* 46 (1986), Nr. 174, S. 439–456

[96] McCulloch, W.S. ; Pitts, W.: A logical calculus of the ideas immanent in nervous activity. In: *Bulletin of Mathematical Biology* 5 (1943), Nr. 4, S. 115–133

[97] Merke, Artur: *Algebraische Analyse von approximativem Reinforcement Lernen*, Universität Osnabrück, Diss., 2005

[98] Michels, J. ; Saxena, A. ; Ng, A.Y.: High speed obstacle avoidance using monocular vision and reinforcement learning. In: *Proc. of the 22nd International Conference on Machine Learning*, 2005, S. 593–600

[99] Minsky, M.: Steps toward artificial intelligence. In: *Proc. of the Instituite of Radio Engineers* 49 (1961), S. 8–30

[100] Monahan, G.E.: A survey of partially observable Markov decision processes: Theory, models, and algorithms. In: *Management Science* 28 (1982), Nr. 1, S. 1–16

[101] Moore, A.W.: Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In: *Proc. of the Eighth International Workshop on Machine Learning*, 1991, S. 333–337

[102] Moore, A.W. ; Atkeson, C.G.: The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In: *Machine Learning* 21 (1995), Nr. 3, S. 199–233

[103] Munos, R. ; Moore, A.: Variable resolution discretization for high-accuracy solutions of optimal control problems. In: *International Joint Conference on Artificial Intelligence*, 1999

[104] Munos, R. ; Moore, A.: Variable resolution discretization in optimal control. In: *Machine Learning* 49 (2002), Nr. 2, S. 291–323

[105] Müller, H. ; Lauer, M. ; Hafner, R. ; Lange, S. ; Merke, A. ; Riedmiller, M.: Making a robot learn to play soccer using reward and punishment. In: *KI 2007* Bd. 4667, Springer, 2007, S. 220–234

[106] Nair, V. ; Hinton, G.E.: 3-d object recognition with deep belief nets. In: *Advances in Neural Information Processing Systems* 22 (2009)

[107] Nene, S.A. ; Nayar, S.K. ; Murase, H.: Columbia object image library (COIL-100). 1996. – Forschungsbericht

[108] Ogino, M. ; Katoh, Y. ; Aono, M. ; Asada, M. ; Hosoda, K.: Reinforcement learning of humanoid rhythmic walking parameters based on visual information. In: *Advanced Robotics* 18 (2004), Nr. 7, S. 677–697

[109] Olshausen, B.A. ; Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. In: *Nature* 381 (1996), S. 607–609

[110] Opršal, I. ; Zahradník, J.: Elastic finite-difference method for irregular grids. In: *Geophysics* 64 (1999), S. 240–250

[111] Ormoneit, D. ; Glynn, P.: Kernel-based reinforcement learning in average-cost problems: An application to optimal portfolio choice. In: *Advances in neural information processing systems* (2001), S. 1068–1074

[112] Ormoneit, D. ; Glynn, P.: Kernel-based reinforcement learning in average-cost problems. In: *IEEE Transactions on Automatic Control* 47 (2002), Nr. 10, S. 1624–1636

[113] Ormoneit, D. ; Sen, Ś.: Kernel-based reinforcement learning. In: *Machine Learning* 49 (2002), Nr. 2, S. 161–178

[114] Ostrovsky, R. ; Rabani, Y. ; Schulman, L.J. ; Swamy, C.: The effectiveness of Lloyd-type methods for the k-means problem. In: *Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006, S. 165–176

[115] Peters, J. ; Schaal, S.: Policy Gradient Methods for Robotics. In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006

[116] Pomerleau, D.A.: *ALVINN: an autonomous land vehicle in a neural network.* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1989

[117] Pomerleau, D.A.: Neural network based autonomous navigation. In: *Vision and Navigation. The Carnegie Mellon Navlab* (1990), S. 83–93

[118] Pomerleau, Dean: Neural Network Vision for Robot Driving. In: Arbib, M. (Hrsg.): *The Handbook of Brain Theory and Neural Networks.* 1995

[119] Press, W.H. ; Teukolsky, S.A. ; Vetterling, W.T. ; Flannery, B.P.: *Numerical recipes: the art of scientific computing.* Cambridge University Press, 2007

[120] Provost, J. ; Kuipers, B.J. ; Miikkulainen, R.: Self-organizing perceptual and temporal abstraction for robot reinforcement learning. In: *AAAI-04 workshop on learning and planning in markov processes*, 2004, S. 79–84

[121] Provost, J. ; Kuipers, B.J. ; Miikkulainen, R.: Self-organizing distinctive state abstraction using options. In: *Proc. of the 7th International Conference on Epigenetic Robotics*, 2007

[122] Puterman, M.L. ; Shin, M.C.: Modified Policy Iteration Algorithms for Discounted Markov Decision Problems. In: *Management Science* 24 (1978), Nr. 11, S. 1127–1137

[123] Raina, R. ; Madhavan, A. ; Ng, A.Y.: Large-scale deep unsupervised learning using graphics processors. In: *Proc. of the 26th International Conference on Machine Learning*, 2009

[124] Ranzato, M.A. ; Huang, Fu J. ; Boureau, Y.-L. ; LeCun, Yann: Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07.*, 2007

[125] Reynolds, S.I.: Decision boundary partitioning: Variable resolution model-free reinforcement learning. School of Computer Science, University of Birmingham, TC CSRP-99-15, 1999. – Forschungsbericht

[126] Reynolds, Stuart I.: Adaptive Resolution Model-Free Reinforcement Learning: Decision Boundary Partitioning. In: *Proc. of the 17th International Conference on Machine Learning*, 2000, S. 783–790

[127] Riedmiller, M.: Advanced supervised learning in multi-layer perceptrons–From backpropagation to adaptive learning algorithms. In: *Computer Standards & Interfaces* 16 (1994), Nr. 3, S. 265–278

[128] Riedmiller, M.: *Selbständig lernende neuronale Steuerungen*, Universität Karlsruhe, Diss., 1996

[129] Riedmiller, M.: High quality thermostat control by reinforcement learning-A case study. In: *Proc. of the Conald Workshop*, 1998, S. 1–2

[130] Riedmiller, M.: Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In: *Proc. of the European Conference on Machine Learning*, Springer, 2005

[131] Riedmiller, M. ; Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: *Proceedings of the IEEE international conference on neural networks* Bd. 1993 San Francisco: IEEE, 1993, S. 586–591

[132] Riedmiller, M. ; Hafner, R. ; Lange, S. ; Lauer, M.: Learning to dribble on a real robot by success and failure. In: *Proc. of the IEEE International Conference on Robotics and Automation*, 2008, S. 2207–2208

[133] Riedmiller, M. ; Montemerlo, M. ; Dahlkamp, H.: Learning to drive a real car in 20 minutes. In: *Proc. of the International Conference on Frontiers in the Convergence of Bioscience and Information Technologies (FBIT)*, 2007

[134] Riedmiller, Martin ; Gabel, Thomas ; Hafner, Roland ; Lange, Sascha: Reinforcement learning for robot soccer. In: *Autonomous Robots* 27 (2009), 07, Nr. 1, S. 55–73

[135] Robbins, H. ; Monro, S.: A stochastic approximation method. In: *The Annals of Mathematical Statistics* 22 (1951), Nr. 3, S. 400–407

[136] Rojas, R.: *Theorie der neuronalen Netze.* Springer, 1993

[137] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological review* 65 (1958), Nr. 6, S. 386–408

[138] Roweis, S. T. ; Saul, L. K.: Nonlinear dimensionality reduction by locally linear embedding. In: *Science* 290 (2000), S. 2323–2326

[139] Rumelhart, D. E. ; Hinton, G.E. ; Williams, R. J.: Learning internal representations by error propagation. In: *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1* (1986), S. 318–362

[140] Rumelhart, D.E. ; Hinton, G.E. ; Williams, R.J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), Nr. 6088, S. 533–536

[141] Rumelhart, D.E. ; McClelland, J.L.: *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*. MIT Press Cambridge, MA, USA, 1986

[142] Russell, S.J. ; Norvig, P. ; Canny, J.F. ; Malik, J. ; Edwards, D.D.: *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, NJ, 1995

[143] Rust, J.: Using randomization to break the curse of dimensionality. In: *Econometrica: Journal of the Econometric Society* (1997), S. 487–516

[144] Salakhutdinov, R.R. ; Hinton, G.E.: Learning a nonlinear embedding by preserving class neighbourhood structure. In: *AI and Statistics*, 2007

[145] Salakhutdinov, R.R. ; Mnih, A. ; Hinton, G.E.: Restricted Boltzmann machines for collaborative filtering. In: *Proc. of the 24th international conference on Machine learning*, 2007, S. 791–798

[146] Samuel, A.L.: Some studies in machine learning using the game of checkers. In: *IBM Journal of Research and Development* 3 (1957), S. 210–219

[147] Schmid, C. ; Mohr, R.: Local grayvalue invariants for image retrieval. In: *IEEE transactions on pattern analysis and machine intelligence* 19 (1997), Nr. 5, S. 530–535

[148] Schneiberg, S. ; Sveistrup, H. ; McFadyen, B. ; McKinley, P. ; Levin, M.F.: The development of coordination for reach-to-grasp movements in children. In: *Experimental Brain Research* 146 (2002), Nr. 2, S. 142–154

[149] Singh, S. ; Jaakkola, T. ; Jordan, M.I.: Reinforcement learning with soft state aggregation. In: *Advances in neural information processing systems* (1995), S. 361–368

[150] Singh, S.P. ; Sutton, R.S.: Reinforcement learning with replacing eligibility traces. In: *Machine learning* 22 (1996), Nr. 1, S. 123–158

[151] Smith, A.J.: Applications of the self-organising map to reinforcement learning. In: *Neural Networks* 15 (2002), Nr. 8-9, S. 1107–1124

[152] Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. In: *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: foundations* (1986), S. 194–281

[153] Springenberg, J. T.: *Machine Learning on massively parallel architectures - a case study*. Bachelor Thesis, Universität Osnabrück, 2009

[154] Steinhaus, H.: Sur la division des corp materiels en parties. In: *Bull. Acad. Polon. Sci* 1 (1956), S. 801–804

[155] Sutton, R.S.: Generalization in reinforcement learning: Successful examples using sparse coarse coding. In: *Advances in neural information processing systems* (1996), S. 1038–1044

[156] Sutton, R.S. ; Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, 1998

[157] Tavazoie, S.F. ; Reid, R.C.: Diverse receptive fields in the lateral geniculate nucleus during thalamocortical development. In: *Nature Neuroscience* 3 (2000), S. 608–616

[158] Taylor, G.W. ; Hinton, G.E.: Factored conditional restricted Boltzmann Machines for modeling motion style. In: *Proc. of the 26th International Conference on Machine Learning*, 2009

[159] Taylor, G.W. ; Hinton, G.E. ; Roweis, S.T.: Modeling human motion using binary latent variables. In: *Advances in Neural Information Processing Systems*, 2007

[160] Tesauro, G.: Practical issues in temporal difference learning. In: *Machine learning* 8 (1992), Nr. 3, S. 257–277

[161] Tesauro, G. ; Sejnowski, T.: A Parallel Network that Learns to Play Backgammon. In: *Artificial Intelligence* 39 (1989), Nr. 3, S. 357–390

[162] Thrun, S.: The role of exploration in learning control. In: *Handbook of intelligent control: Neural, fuzzy and adaptive approaches* (1992), S. 527–559

[163] Thrun, S.B.: Efficient exploration in reinforcement learning. In: *Technical Report CMU-CS-92-102, Carnegie Mellon University, School of Computer Science* (1992)

[164] Tian, N. ; Copenhagen, D.R.: Visual Stimulation Is Required for Refinement of ON and OFF Pathways in Postnatal Retina. In: *Neuron* 39 (2003), Nr. 1, S. 85–96

[165] Toschanowitz, K. Tluk v.: *Relevance Determination in Reinforcement Learning*, Universität Bielefeld, Diss., 2008

[166] Toschanowitz, K. Tluk v. ; Hammer, B. ; Ritter, H.: Relevance Determination in Reinforcement Learning. In: *Proc. of European Symposium on Artificial Neural Networks*, 2005

[167] Touzet, CF: Neural reinforcement learning for behaviour synthesis. In: *Robotics and Autonomous Systems* 22 (1997), Nr. 3, S. 251–281

[168] Vincent, P. ; Larochelle, H. ; Bengio, Y. ; Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: *Proc. of the 25th international conference on Machine learning*, 2008, S. 1096–1103

[169] Watkins, C.: *Learning from delayed rewards*, King's College, Cambridge, UK, Diss., 1989

[170] Watkins, C. ; Dayan, P.: Q-Learning. In: *Machine Learning* 8 (1992), S. 279–292

[171] Wehenkel, L. ; Glavic, M. ; Ernst, D.: New Developments in the Application of Automatic Learning to Power System Control. In: *Proc. of the 15th Power Systems Computation Conference (PSCC05)*, 2005

[172] Werbos, P.J.: *Beyond regression: New tools for prediction and analysis in the behavioral sciences*, Harvard University, Diss., 1974

[173] Widrow, B. ; Hoff Jr., M.E.: Adaptive switching circuits. 1960. In: *IRE WESCON Convention Record* Bd. 4, 1960, S. 96–104

[174] Wiskott, L. ; Sejnowski, T.J.: Slow feature analysis: Unsupervised learning of invariances. In: *Neural computation* 14 (2002), Nr. 4, S. 715–770

[175] Wyss, R. ; König, P. ; Verschure, P. F. M. J.: A Model of the Ventral Visual System Based on Temporal Stability and Local Memory. In: *PLoS Biology* 4 (2006), Nr. 5

[176] Zell, Andreas: *Simulation neuronaler Netze*. Oldenbourg Verlag, 1997

[177] Zhu, W. ; Levinson, S.: Vision-based reinforcement learning for robot navigation. In: *Proc. of the International Joint Conference on Neural Networks*, 2001