Aus dem Institut für Neuro- und Bioinformatik
der Universität zu Lübeck

Direktor:
Herr Professor Dr. rer. nat. Thomas Martinetz

Steigerung der Informationseffizienz im Reinforcement-Learning

Inauguraldissertation
zur
Erlangung der Doktorwürde
der Universität zu Lübeck
- Aus der Technisch-Naturwissenschaftlichen Fakultät -

Vorgelegt von
Herrn Daniel Schneegaß
aus Rathenow

Lübeck, 2008

1. Berichterstatter: Herr Professor Dr. rer. nat. Th. Martinetz

2. Berichterstatter: Herr Priv.-Doz. Dr.-Ing. Th. Runkler

Tag der mündlichen Prüfung: 11.12.2008

zum Druck genehmigt. Lübeck, den 11.12.2008

gez. Herr Professor Dr. rer. nat. Jürgen Prestin
- Dekan der Technisch-Naturwissenschaftlichen Fakultät -

Zusammenfassung

Die vorliegende Arbeit befasst sich mit Reinforcement-Learning-Verfahren zur Lösung von Optimalsteuerungs- und -regelungsproblemen im Batch-Learning-Kontext. In den meisten der hier betrachteten Fälle liegen also endliche Datensätze von Beobachtungen vor, die mit in der Regel nur begrenzt beeinflussbaren Explorationsstrategien gewonnen wurden. Unser Ansatz unterscheidet sich damit in einem wesentlichen Punkt vom Fokus der meisten Reinforcement-Learning-Verfahren, die explorativ Daten sammeln und darauf aufbauend Strategien entwickeln. Vorrangiges Ziel dieser Arbeit besteht daher in der Entwicklung und Analyse von Maßnahmen zur Steigerung der Informationseffizienz, also bestmöglicher Ausnutzung verfügbarer Informationen. Dazu zählen neben der begrenzten Zahl an Beobachtungen auch a priori Annahmen.

Die Arbeit umfasst zwei wesentliche Beiträge. Der erste Beitrag besteht in der Einführung sowie umfassenden Darstellung und Untersuchung des Rewards-Regression-Ansatzes, der sich das Prinzip der strukturellen Risiko-Minimierung im Kontext von Reinforcement-Learning zu Nutze macht. Diese hier eingeführten Verfahren arbeiten auf der Grundlage neuronaler Netze und Kernel-Machines. Dabei wird der Anspruch erhoben, eine möglichst breite und allgemeine Problemklasse für unterschiedliche Optimalitätskriterien lösen zu können. Dazu bedienen wir uns einer Umformulierung von Reinforcement-Learning-Problemen in Regressionsprobleme, so dass theoretische und praktische Erkenntnisse aus der Theorie der Regression angewendet werden können. Neben klassischen Regularisierungstechniken in neuronalen Netzen und jüngeren, statistisch motivierten Verfahren, die in Kernel-Machines und Support-Vector-Machines zum Einsatz kommen, werden auch Regularisierungstechniken betrachtet, die in dynamischen Systemen, besonders zur Behandlung partiell beobachtbarer Probleme von Interesse sind. Ferner untersuchen wir Konsistenz und Erwartungstreue der vorgestellten Verfahren sowie die weitere Steigerung der Generalisierungsleistung durch parallele Regularisierung von Bewertungsfunktion und Policy.

Der zweite Beitrag besteht in der Kombination von Unsicherheitsbetrachtungen mit Reinforcement-Learning. Dazu wird die Unsicherheit der initialen Beobachtungen bestimmt und durch die Lernverfahren propagiert, so dass schließlich, zu Gunsten der Qualität der Lösung, die Unsicherheit der Policy ermittelt und berücksichtigt werden kann. Diese Unsicherheiten werden im Bayes'schen Kontext gewonnen. Daher stellt dieser Ansatz eine zur strukturellen Risiko-Minimierung alternative Methode dar, die Generalisierungsleistung zu verbessern, bei dem die Wahl der Struktur durch die Wahl eines Priors ersetzt wird. Darauf aufbauend führen wir das Konzept der Sicherheitsoptimalität ein. Die sicherheitsoptimale Policy erreicht eine maximal mögliche Mindestperformance mit zuvor festgelegter Wahrscheinlichkeit. Damit eröffnet sich auch ein breites Anwendungsfeld im Bereich der Qualitätssicherung, das sehr eng mit der Problematik der Informationseffizienz verwoben ist. Darüber hinaus erweitern wir die hier vorgestellten Techniken zur Anwendung auf Funktionsapproximatoren sowie für verschiedene statistische Paradigmen und unterschiedliche Optimalitätskriterien.

Die Methoden aus beiden Beiträgen werden schließlich auf verschiedenen Benchmarks getestet und auf unterschiedlichen Anwendungen um industriell eingesetzte Gasturbinen erprobt.

Danksagung

Die vorliegende Arbeit ist in einer Kooperation des Instituts für Neuro- und Bioninformatik der Universität zu Lübeck und der Siemens AG im Rahmen verschiedener Forschungsprojekte in der Abteilung Corporate Technology, Information & Communications, Learning Systems entstanden. Dadurch haben sowohl Know-How, Erfahrung als auch Produkte des Siemens-Fachzentrums wesentlich zur Entstehung der Arbeit beigetragen.

In erster Linie danke ich Herrn Professor Dr. Thomas Martinetz für die Betreuung der Arbeit und die mir zuteil gewordene Unterstützung und Beratung. Ich danke Herrn Professor Dr. Bernd Schürmann und seinem Nachfolger Herrn Dr. Thomas Runkler, die mir die Möglichkeit gegeben haben, als Mitglied des ausgezeichneten Forscherteams des Fachzentrums Learning Systems zu arbeiten und die Aufgaben der Abteilung mitzugestalten.

Mein Dank geht an alle Kollegen der Abteilung, insbesondere an Herrn Volkmar Sterzing für seine tatkräftige Unterstützung bei der Durchsetzung unserer Technologien für den Einsatz in Bereichsprojekten, wodurch sich zahlreiche Gelegenheiten ergaben, die theoretische Arbeit mit praktischen Herausforderungen erfolgreich zu kombinieren. Hervorheben möchte ich auch alle Kollegen und Doktoranden seines Kompetenzfeldes wegen der fruchtbaren Zusammenarbeit auf Diskussions- und Entwicklungsebene.

Mit den Kollegen Herr Dieter Bogdoll, Herr Dr. Ralph Grothmann, Herr Alexander Hans, Herr Anton Schäfer, Herr Dr. Hans-Georg Zimmermann sowie auf der Power-Generation-Seite Herr Dr. Hans-Gerd Brummel, Herr Ron Rudolph und Herr Jatinder Singh habe ich erfolgreich an Projekten und Fachbeiträgen zusammenarbeiten können. Mit vielen wertvollen Hinweisen und Diskussionsbeiträgen konnten mich darüber hinaus die Kollegen Herr Bernhard Lang, Herr Dr. Peter Mayer, Herr Dr. Michael Metzger, Herr Dr. Dragan Obradovic, Herr Dr. Paul-Theo Pilgram und Herr Dr. Volker Tresp unterstützen.

Mein besonderer Dank gilt Herrn Dr. Steffen Udluft, der mich fachlich während der gesamten Zeit betreut und mir zahlreiche Einblicke in die Facetten des Machine-Learnings und der Statistik aus der Sicht eines Physikers gewährt hat. Er unterstützte mich bei all meinen Ideen und Vorstößen und brachte viele Ideen und Anregungen in die Arbeit ein.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Abkürzungsverzeichnis

BPG	Bayesian-Policy-Gradient-Verfahren
DP	Dynamische Programmierung
EKRR	Explicit-Kernel-Rewards-Regression
GPTD	Gaussian-Process-Temporal-Difference-Learning
KBRL	Kernel-basiertes Reinforcement-Learning
KRR	Kernel-Rewards-Regression
LSPI	Least-Squares-Policy-Iteration
MDL	Minimale Beschreibungslänge
MDP	Markov-Entscheidungsprozess
NFQ	Neural-Fitted-Q-Iteration
NRR	Neural-Rewards-Regression
PAC	Probably-Approximately-Correct-Learning
PGNRR	Policy-Gradient-Neural-Rewards-Regression
PGRL	Policy-Gradient-Reinforcement-Learning
POMDP	Partiell beobachtbarer Markov-Entscheidungsprozess
RCNN	Recurrent-Control-Neural-Network
RL	Reinforcement-Learning
RNN	Rekurrentes neuronales Netz
RNRR	Recurrent-Neural-Rewards-Regression
RPGNRR	Recurrent-Policy-Gradient-Neural-Rewards-Regression
SVM	Support-Vector-Machine
TD-Learning	Temporal-Difference-Learning
UPRL	Unsicherheitspropagation für Reinforcement-Learning
VC-Dimension	Vapnik-Chervonenkis-Dimension

Abb(n).	Abbildung(en)	i.d.R.	in der Regel
Abschn.	Abschnitt(e)/	i.e.S.	im engeren Sinne
	Unterabschnitt(e)	i.S.	im Sinne
Alg.	Algorithmus	i.w.S.	im weiteren Sinne
Beob.	Beobachtung	Kap.	Kapitel
bzgl.	bezüglich	o.B.d.A.	ohne Beschränkung
bzw.	beziehungsweise		der Allgemeinheit
entf.	entfällt	Tab(n).	Tabelle(n)
Erwart.	Erwartungswert	u.a.	unter anderem
gesch.	geschätzt	z.B.	zum Beispiel
Gl(n).	Gleichung(en)	z.T.	zum Teil

Notationsverzeichnis

a	Aktion
a'	Folgeaktion
A	Aktionsraum
$\tilde{A}_i$	Menge von lokal optimalen Aktionen am $i$ten Folgezustand
b	Bias
c	Konzept
C	Konzeptklasse
C^m	Kovarianzmatrix in der $m$ten Iteration
$\mathbf{Cov}((P,R),(P,R))$	Kovarianzmatrix der initialen Schätzer für die Transitionswahrscheinlichkeiten und Rewards
$\mathbf{Cov}((Q^m,P,R),(Q^m,P,R))$	Kovarianzmatrix zwischen Q-Funktion, Transitionswahrscheinlichkeiten und Rewards in der $m$ten Iteration
$\mathbf{Cov}(Q^*,Q^*)$	Kovarianzmatrix der optimalen Q-Funktion
C_Q	Konzeptklasse von Q-Funktionen
C_π	Konzeptklasse von Policies
$d(c,h)$	Abstand von Konzept c zu Hypothese h
$\mathbf{diag}(A)$	Diagonale der Matrix A als Vektor
D^m	Jacobi-Matrix in der $m$ten Iteration
$D(s,a)$	Gesamtertrag von Zustand s bei Aktion a
D^*	Jacobi-Matrix nach Konvergenz der Bellman-Iteration
$\mathbf{e}$	Vektor mit aufsteigenden ganzzahligen Einträgen, diskreter Erwartungswertoperator
$\mathrm{err}(f)$	Fehler der Funktion f
$\mathbf{E}_\mathbf{s}^\pi$	Erwartungswert über Zustände $\mathbf{s}$ bei Anwendung der Policy π
$\mathbf{E}_\mathbf{x}$	Erwartungswert über $\mathbf{x}$
$\mathbf{E}(\theta\|Z,H)$	Erwartungswert der Hypothesen-Parameter θ für gegebenen Hypothesenraum H und Beobachtungen Z
$f(\mathbf{x},\alpha)$	Funktionsapproximator für Argumente $\mathbf{x}$ mit Parametern α
F	Operator, der auf einen Zustand wirkt und in einen internen Markov-Zustand abbildet
$F(L_i)$	Fehlerfunktion zur Weiterverarbeitung einer Verlustfunktion
G	Operator, der auf eine Aktion wirkt und in einen dynamik-internen Zustand abbildet

h	Hypothese
H	Operator, der auf einen internen Markov-Zustand wirkt und in einen dynamik-internen Zustand abbildet
H	Hypothesenraum
H_L	Hypothesenraum von Verlustfunktionen
H_Q	Hypothesenraum von Q-Funktionen
H_π	Hypothesenraum von Policies
J	Operator, der im Vergangenheits- und Gegenwartsteil auf einen dynamik-internen Zustand wirkt und in einen internen Markov-Zustand abbildet
K	Kernel-Matrix zwischen Zuständen
K'	Kernel-Matrix zwischen Zuständen und Folgezuständen
K''	Kernel-Matrix zwischen Folgezuständen und Zuständen
K'''	Kernel-Matrix zwischen Folgezuständen und Folgezuständen
K'_i	Kernel-Matrix über Folgezustands-Aktions-Paare, für die die auszuwertenden Folgezustands-Aktions-Paare so ersetzt werden, dass die Aktionen durch die mit der Nummer i substituiert werden
$K((s,a),(s_i,a_i))$	Kernel für Zustands-Aktions-Paare
$\tilde{K}((s,a,s'),(s_i,a_i,s'_i))$	Gesamter Kernel für Zustands-Aktions-Folgezustands-Tripel
$K(s,s_i)$	Kernel für Zustände
$\hat{K}(s,s_i)$	Kernel für Zustände, der auf einen Kernel für Zustands-Aktions-Paare zurückgeführt wird
$K(\mathbf{x}_i,\mathbf{x})$	Kernel-Funktion für Beobachtung $\mathbf{x}$ und Referenzbeobachtung $\mathbf{x}_i$
l	Anzahl der Beobachtungen
L	Erwartungstreuer Schätzer für das deterministische Bellman-Residuum für eine Verteilung von Beobachtungen
$L_{\text{aux}}(s,a)$	Auxiliared-Bellman-Residuum für eine Beobachtung
$L_C(s,a)$	Konsistenter Schätzer für das Bellman-Residuum für eine Beobachtung
$L_D(s,a)$	Erwartungstreuer Schätzer für das Bellman-Residuum für eine Beobachtung unter Berücksichtigung zweier Schätzer für den Q-Wert am Folgezustand
$L'(f,g)$	Abstandsmaß zweier Q-Funktionen, das an das Bellman-Residuum angelehnt ist

$L(s,a)$	Erwartungstreuer Schätzer für das deterministische Bellman-Residuum für eine Beobachtung
$\hat{L}(s,a)$	Komponente, die zum deterministischen Bellman-Residuum addiert wird und damit zum Auxiliared-Bellman-Residuum führt
$L_V(s,a)$	Erwartungstreuer Schätzer für das Bellman-Residuum für eine Beobachtung unter Berücksichtigung der Kenntnis der Varianz des Gesamtertrags
$L(\mathbf{w}, b, \alpha)$	Lagrange-Funktional für Gewichtsvektor $\mathbf{w}$, Bias b und Lagrange-Koeffizienten α
$L(x)$	Verlustfunktion für Beobachtung x
$L(y, y')$	Verlust-Funktion für wahren Wert y und Schätzer y'
M	Operator, der auf einen dynamik-internen Zustand wirkt und in einen Ausgabevektor abbildet
$n_{s,a}$	Anzahl der beobachteten Transitionen ausgehend vom entsprechenden Zustands-Aktions-Paar
$n_{s'\|s,a}$	Anzahl der beobachteten Transitionen ausgehend vom entsprechenden Zustands-Aktions-Paar, die in einen bestimmten Folgezustand führen
N	Operator, der im Zukunftsteil auf einen dynamik-internen Zustand wirkt und in einen internen Markov-Zustand abbildet
$N_a(s)$	Darstellung der Q-Funktion als ein neuronales Netz mit dem Zustand als Eingabegröße und der Aktion als Parameter
NOX	Stickstoffoxidausstoß der Turbine
$N(s,a)$	Darstellung der Q-Funktion als ein neuronales Netz mit Zustand und Aktion als Eingabegrößen
p_{all}	Wahrscheinlichkeit für das gleichzeitige Unterschreiten der Minimalperformance für alle Zustands-Aktions-Paare
p_{global}	Wahrscheinlichkeit für das Unterschreiten der Minimalperformance für alle Zustands-Aktions-Paare hinsichtlich eines bestimmten Kriteriums
p_{least}	Wahrscheinlichkeit für das Unterschreiten der Minimalperformance für mindestens ein Zustands-Aktions-Paar
p_{single}	Wahrscheinlichkeit für das Unterschreiten der Minimalperformance für ein einzelnes Zustands-Aktions-Paar

$P(A)$	A priori Wahrscheinlichkeit von A
$P(B\|A)$	A posteriori Wahrscheinlichkeit von B gegeben A
PEL	Turbinen-Leistung
$P_R(s, a, s', r)$	Wahrscheinlichkeit für die Vereinnahmung des Rewards r beim Übergang von s über a nach s'
$R(s, a, s')$	Reward beim Übergang von s über a nach s'
$P(s'\|s, a)$	Bedingte Transitionswahrscheinlichkeit von s über a nach s'
$P_T(s, a, s')$	Transitionswahrscheinlichkeit von s über a nach s'
$P(\mathbf{x})$	Wahrscheinlichkeit für die Beobachtung $\mathbf{x}$
$P(\mathbf{x}, y)$	Gemeinsame Wahrscheinlichkeit von Beobachtung $\mathbf{x}$ und Ergebnis y
$P(\mathbf{z}\|Z, H)$	Wahrscheinlichkeit für neues Datum $\mathbf{z}$ für gegebenen Hypothesenraum H und Beobachtungen Z
$P(\theta\|Z, H)$	Wahrscheinlichkeit für Hypothesen-Parameter θ für gegebenen Hypothesenraum H und Beobachtungen Z
$P_\pi(s)$	Stationäre Wahrscheinlichkeit von s bei Anwendung von π
Q	Vektor mit Q-Funktions-Einträgen an den beobachteten Zuständen
Q'	Vektor mit Q-Funktions-Einträgen an den beobachteten Folgezuständen
Q^m	Q-Funktion in der $m$ten Iteration
$\hat{Q}_n(s, a)$	Temporärer Schätzer der Q-Funktion bei Anwendung von Eligibility-Traces
$\hat{Q}(s, a)$	Temporärer Schätzer der Q-Funktion
$Q^*(s, a)$	Q-Funktion der optimalen Policy in Zustand s für Aktion a
$Q_u(s, a)$	Garantierte Minimalperformance mit einer bestimmten Wahrscheinlichkeit
$Q^\xi(s, a)$	Performance der bzgl. ξ sicherheitsoptimalen Policy
$Q_u^\xi(s, a)$	Garantierte Minimalperformance mit einer bestimmten Wahrscheinlichkeit für die bzgl. ξ sicherheitsoptimale Policy
$Q^\pi(s, a)$	Q-Funktion der Policy π in Zustand s für Aktion a
$\bar{Q}^\pi(s, a)$	Wahre Performance der Policy π in Zustand s für Aktion a
r	Reward
R	Vektor mit Reward-Einträgen
R_{Bellman}	Bellman-Residuum
RMS	Dynamik der Brennkammer der Turbine

$R(\alpha)$	Risiko-Funktional für Parameter α
s	Zustand
s'	Folgezustand
$s^{(t)}$	Folgezustand nach t Schritten
S	Zustandsraum
S_i	Hypothesenraum innerhalb einer Struktur von Hypothesenräumen
TQ	Bellman-Operator der optimalen Policy auf Q
$T'Q$	Standardabweichung des Gesamtertrags bzgl. Q bei Anwendung der optimalen Policy
$T^\pi Q$	Bellman-Operator der Policy π auf Q
$(T')^\pi Q$	Standardabweichung des Gesamtertrags bzgl. Q bei Anwendung der Policy π
var	Varianzoperator
$\mathbf{v}_i$	Vektor mit erwarteten Folgezuständen eines Markov-Prozesses nach i Iterationen
$\mathbf{v}_+$	Vektor mit den Differenzen zwischen den Q-Werten zweier verschiedener Aktionen, falls der jeweils erste Q-Wert größer ist
$\mathbf{v}_-$	Vektor mit den Differenzen zwischen den Q-Werten zweier verschiedener Aktionen, falls der jeweils zweite Q-Wert größer ist
V	Vektor mit Bewertungsfunktions-Einträgen an den beobachteten Zuständen
V'	Vektor mit Bewertungsfunktions-Einträgen an den beobachteten Folgezuständen
$\mathbf{VCD}(H)$	VC-Dimension für Klassifikationsprobleme des Hypothesenraumes H
$V^\pi(s)$	Bewertungsfunktion der Policy π im Zustand s
$V^*(s')$	Bewertungsfunktion der optimalen Policy im Zustand s
$\mathbf{w}$	Gewichtsvektor
$\hat{\mathbf{w}}$	Gewichtsvektor für interne Zustandsübergänge
$\mathbf{w}_a$	Gewichtsvektor, der durch eine Aktion parametrisiert wird
$\mathbf{x}_i$	Beobachtung
y_i	Funktionswert
$Y(s, a, s')$	Schätzer für die Anwendung des Bellman-Operators auf die Q-Funktion für Zustand, Aktion und Folgezustand

α	Lernrate
α_i	Lagrange-Koeffizient i
γ	Diskontierungsfaktor
Γ	Projektionsoperator
δ	Minimale Fehlerwahrscheinlichkeit
Δ	Gradient, Differenzial
ε	Minimaler Fehler
θ	Parameter der Q-Funktion
κ	Parameter, der den Trade-Off zwischen verschiedenen globalen Sicherheitskriterien gestaltet
λ	Regularisierungsparameter
μ_k	Statistisches Moment $k$ter Ordnung
ξ	Konfidenzlevel
π^m	Policy in der $m$ten Iteration
$\pi(s)$	Deterministische Policy, liefert Aktion im Zustand s
$\pi^*(s)$	Optimale deterministische Policy
$\pi(s,a)$	Stochastische Policy, liefert Wahrscheinlichkeit für die Wahl von Aktion a im Zustand s
$\pi^\xi(s,a)$	Policy, die bzgl. ξ sicherheitsoptimal ist
Π	Aktionsauswahloperator
$\Pi(H_Q)$	Funktion, die auf einen Hypothesenraum von Q-Funktionen wirkt und in einen Hypothesenraum von Policies abbildet, der zurückgegebene Hypothesenraum ist konsistent zur Eingabe in dem Sinne, dass zu jeder in ihm enthaltenen Policy eine in H_Q enthaltende Q-Funktion existiert, die diese Policy hervorbringen kann und umgekehrt
ρ	Parameter zur Gestaltung des Trade-Offs zwischen Fixpunkt der Bellman-Iteration und Minimum des Bellman-Residuums
σ	Standardabweichung
σ	Unsicherheitsoperator
τ	Temperatur
Φ	Feature-Matrix für die Zustände
Φ'	Feature-Matrix für die Folgezustände
$\Phi(s,a)$	Feature-Funktion für Zustands-Aktions-Paare
$\Phi(\mathbf{x})$	Feature-Funktion für Beobachtung $\mathbf{x}$
ω	Parameter der Policy
Ω	Regularisierungsoperator
∂	Partielle Ableitung

Kapitel 1

Einführung

Reinforcement-Learning [107, 13] (RL) ist der Ansatz des maschinellen Lernens (Machine-Learning) zur Lösung von Optimalsteuerungs- und -regelungsproblemen (Optimal Control Problem) und geht zurück auf Richard Bellman [11], der den Begriff der dynamischen Programmierung einführte und auf Optimalsteuerungs- und -regelungsprobleme anwendete. Dynamische Programmierung ist eine allgemeine algorithmische Technik mit einem umfassenden Anwendungsfeld in der Optimierung [16] und zusammen mit dem Machine-Learning wesentliche Grundlage des Reinforcement-Learnings, das als Teilgebiet beider Schwerpunkte aufgefasst werden kann.

Machine-Learning und künstliche Intelligenz wurden ursprünglich durch biologische Nervensysteme motiviert, nach deren Vorbild künstliche neuronale Netze entwickelt wurden. Das Ziel besteht darin, Computer zu konstruieren, die über die Intelligenz eines Menschen verfügen oder diese zumindest simulieren können. Auf Alan Turing geht der sogenannte Turing-Test zurück [113], der die Intelligenz von Computern überprüfen soll. Er empfiehlt das erfolgreiche Absolvieren des Nachahme-Spieles (Imitation Game) durch einen Computer zum Nachweis seiner Intelligenz. Während sich Turing in [113] auf Lernmaschinen bezieht, bestand ein weiteres Ziel darin, einen allgemeinen Problemlöser zu entwickeln. Jedoch erst in den 80er Jahren des 20. Jahrhunderts wurden Computer so leistungsfähig, dass mit diesen Methoden praktische Probleme angegangen werden konnten.

Neue Fachgebiete wie Neuroinformatik, Bioinformatik, Data-Mining etablierten sich, die Sachverhalte von sehr ähnlichen, sich teilweise überschneidenden Themen untersuchen. Als theoretisches Fundament der inhärenten Eigenschaften des Machine-Learnings wird die statistische Lerntheorie (Statistical Learning Theory) aufgefasst, die vor allem begründet wurde durch Vapnik und Chervonenkis [116, 117], aber in ihren Ursprüngen zurückgeht auf Occam und Bayes. Sie ist wiederum ein Teilgebiet der Computational Learning Theory, die zum Beispiel auch Komplexitätseigenschaften von Machine-Learning-Algorithmen untersucht. Einführende und z.T. vertiefende Ausführungen zu diesen Themen finden sich bei McKay [59] und Hastie et al. [40].

Machine-Learning kommt heute bei einer Vielzahl von Aufgabenstellungen der mathematischen Modellierung zum Einsatz, u.a. in den Bereichen Klassifikation, Regression, Dichteschätzung, Steuerung und Regelung sowie Optimierung, im industriellen Umfeld bei Problemstellungen, für die eine vollständige oder zumindest ausreichende Beschreibung der zu Grunde liegenden Phänomene nicht vorliegt. Dies kann etwa bei komplexen maschinellen Anlagen der Fall sein, in der Medizin oder bei marktwirtschaftlichen Fragestellungen. Zwar sind die grundlegenden Gesetzmäßigkeiten oftmals bekannt, jedoch kann eine exakte Herleitung seiner Funktionsweise auf Grund der Komplexität des Gesamtsystems nicht vollzogen werden. In solchen Fällen können mit Hilfe eines Lehrers, der exemplarisch, also an einzelnen Beobachtungen, die Zusammenhänge beschreibt, Datensätze generiert werden, die es den Machine-Learning-Verfahren ermöglichen, die phänomenologischen Zusammenhänge zu erkennen und daraus die richtigen Entscheidungen abzuleiten, also zu lernen. Dieses Vor-

gehen wird auch als überwachtes Lernen (Supervised Learning) bezeichnet, das sich vom unüberwachten Lernen (Unsupervised Learning) gerade darin unterscheidet, dass ein Lehrer zur Verfügung steht. Reinforcement-Learning wird i.d.R. als zwischen diesen beiden Methoden liegend aufgefasst, da der Lehrer nicht die Aufgabe hat, richtige Aktionen vorzuführen, sondern lediglich den Reward, also den aktuellen Nutzen, zu präsentieren. Die Handlungsstrategie soll nur an Hand dieser Informationen gelernt werden.

Ein wichtiges Ziel des Machine-Learnings besteht ferner darin, mit einer begrenzten Zahl an Beobachtungen brauchbare Lösungen anzubieten. Dies ist insbesondere dann von Bedeutung, wenn einzelne Beobachtungen kostbar sind, weil sie Zeit oder Geld erfordern oder der Lehrer nicht beliebig lange zur Verfügung steht. Obwohl Reinforcement-Learning auf Machine-Learning beruht, sind hier noch viele Fragen offen, deren Entsprechungen bei Klassifikations- und Regressionsproblemen bereits beantwortet sind. Auch bei der praktischen Anwendung wird ein solches Bild gezeichnet. Die meisten etablierten Methoden legen den Fokus bisher auf andere Probleme als eine möglichst dateneffiziente, auf hohe Generalisierung ausgerichtete Lösung zu bestimmen. Das vermutlich wichtigste von ihnen wird mit dem Begriff Explorations-Ausbeutungs-Dilemma (Exploration-Exploitation-Dilemma) bezeichnet. Denn Reinforcement-Learning wird meist angewendet im Kontext von interaktivem oder Online-Lernen. Da interaktives Lernen hier per Konstruktion stets aktives Lernen (Active Learning) bedeutet, kann das regelnde System bereits während des Lernens beeinflussen, welche Daten zum Lernen herangezogen werden sollen. Diese Fragestellung wollen wir in dieser Arbeit bewusst aussparen und uns darauf konzentrieren, beobachtete Trajektorien informationseffizient zur Adaption von möglichst guten (near-optimal) Regelungssystemen (Controller) zu nutzen.

Ziel dieser Arbeit ist es daher, vorliegende Reinforcement-Learning-Verfahren so zu modifizieren und zu erweitern, dass sie möglichst dateneffizient arbeiten. Dabei sollen, als erster Beitrag dieser Arbeit, Reinforcement-Learning-Probleme so in Regressionsprobleme überführt werden, dass Methoden des Machine-Learnings zur Regression angewendet und theoretische Ergebnisse der statistischen Lerntheorie genutzt werden können. Es sollen dabei auch eine möglichst breite Problemklasse abgedeckt werden und unterschiedliche lerntheoretische Methoden und Paradigmen zur Anwendung kommen. So wenden wir die sogenannte Rewards-Regression sowohl auf neuronale Netze als auch auf Kernel-Machines an, die sich in wesentlichen Punkten unterscheiden. In beiden Ansätzen wird untersucht, wie eine optimale Ausnutzung verfügbarer Daten erzielt und unterschiedliche Regularisierungstechniken angewendet werden können. Wir diskutieren, inwieweit die Anwendung kontinuierlicher Aktionen möglich ist und erweitern die vorgestellten Verfahren für den Einsatz im industriellen Umfeld. In diesem Zusammenhang werden auch einige theoretische Beobachtungen zur Dateneffizienz im Reinforcement-Learning erläutert. So untersuchen wir Konsistenz und Erwartungstreue von wichtigen Optimalitätskriterien im Batch-Reinforcement-Learning und gehen auf die Möglichkeit der verschränkten Regularisierung von Q-Funktion und Policy ein. Schließlich zeigen wir, dass die Generalisierungsleistung weiter gesteigert werden kann gegenüber entsprechender Standard-Verfahren, auch gegenüber hier zuvor vorgestellter Verfahren für Batch-Reinforcement-Learning.

Der zweite Beitrag dieser Arbeit beschäftigt sich mit einem anderen Aspekt der Dateneffizienz. Bei sehr wenigen verfügbaren Beobachtungen und auf Grund der Stochastizität allgemeiner MDP ist es von großer Bedeutung, die statistische Unsicherheit der Beobachtungen mit einzubeziehen. Daher entwickeln wir Methoden, deren durchschnittliche ermittelte Policy für eine wichtige Klasse von Optimalsteuerungs- und -regelungsproblemen eine höhere Performance erzielt als ein entsprechendes Standard-Verfahren, mindestens jedoch ihre Varianz reduziert und daher etwa in sicherheitskritischen Umgebungen von Bedeutung ist. Ein wesentlicher Aspekt ist hier auch die Qualitätssicherung. Denn das Berücksichtigen von Unsicherheit im Reinforcement-Learning erlaubt, eine Mindestperformance mit zuvor festgelegter Wahrscheinlichkeit zu garantieren und diese sogar zu maximieren. Während im ersten Beitrag das Prinzip der strukturellen Risiko-Minimierung im Vordergrund steht, wird hier das Bayes'sche Paradigma das Fundament bieten. Wir zeigen, dass die Methode sehr

allgemein einsetzbar ist und erweitern sie auf komplexe Reinforcement-Learning-Verfahren. Darüber hinaus diskutieren wir die Möglichkeiten einer umfassenden Ausnutzung aller verfügbaren statistischen Informationen.

Die Beiträge unterscheiden sich also in ihren grundsätzlichen Herangehensweisen. Daher wird ein Vergleich und eine Gegenüberstellung dieser Paradigmen vollzogen. Ihnen ist gemeinsam, dass sie die Steigerung der Informationseffizienz im Reinforcement-Learning zum Ziel haben und erreichen gerade dadurch und durch ihre individuellen Schwerpunkte die Einsatzfähigkeit im industriellen Umfeld.

Im nächsten Teil (Teil I) wird das allgemeine Optimalregelungsproblem definiert und beschrieben, auf welche Weise Reinforcement-Learning es einer Lösung zuführt (Kap. 2). Wir stellen einige etablierte Verfahren vor. Im Folgenden werden in Kap. 3 wesentliche Elemente der statistischen Lerntheorie und wichtige Beiträge zur Regression vorgestellt. Dabei spielen neben neuronalen Netzen auch Kernel-Verfahren, die im Reinforcement-Learning zur Zeit etabliert werden, Bayes'sche und weitere statistische Methoden eine Rolle. In Kap. 4 werden wir schließlich aktuelle Beiträge beschreiben, die den Stand der Technik im Kontext von Dateneffizienz im Reinforcement-Learning widerspiegeln.

Im Anschluss daran wird der erste Hauptbeitrag dieser Arbeit erläutert, der sogenannte Rewards-Regression-Ansatz (Teil II). Wir werden in Kap. 5 einige prinzipielle Eigenschaften beschreiben und die Kernel-Rewards-Regression vorstellen (Kap. 6), die auf Algorithmen aus Kap. 4 aufbaut. Wir stellen ferner Methoden vor, um Support-Vector-Machines bzw. entsprechende Regularisierungstechniken einzusetzen und erläutern einen allgemeineren und direkten Ansatz, der auf quadratischer Programmierung beruht. Symmetrisch gehen wir schließlich auf die Neural-Rewards-Regression ein (Kap. 7), die ebenfalls auf Arbeiten aus Kap. 4 aufbaut. Wir diskutieren Erweiterungsmöglichkeiten zur Behandlung kontinuierlicher Aktionen und die Anwendbarkeit in partiell beobachtbaren Umgebungen. In diesem Zusammenhang werden rekurrente neuronale Netze eine zentrale Rolle spielen. Wir befassen uns mit theoretischen Untersuchungen zur Steigerung der Dateneffizienz und zeigen in Kap. 8 praktische Ergebnisse auf Benchmarks und industriellen Anlagen.

Der zweite Hauptbeitrag dieser Arbeit wird in Teil III vorgestellt. Nach der Skizzierung der physikalischen Motivation und der Einführung der Unsicherheitspropagation in Kap. 3, wird diese Technik auf die Bellman-Iteration angewendet (Kap. 9). Wir diskutieren die Anwendung unterschiedlicher statistischer Paradigmen, insbesondere die frequentistische und die Bayes'sche Herangehensweise. Im Anschluss daran führen wir das Konzept der Sicherheitsoptimalität ein, das zu einer maximalen garantierten Mindestperformance mit zuvor festgelegter Wahrscheinlichkeit führt und daher zur Qualitätssicherung herangezogen werden kann. Wir erläutern schließlich verschiedene Anwendungsmöglichkeiten dieser Methode und zeigen, wie auf einer wichtigen Problemklasse die Informationseffizienz gesteigert werden kann. Die Methode wird in Kap. 10 mit weiteren etablierten Reinforcement-Learning-Verfahren kombiniert. Dies führt zu einer weitreichenden praktischen Anwendbarkeit. Schließlich untersuchen wir eine verallgemeinerte globale Form der Sicherheitsoptimalität und analysieren den Einfluss weiterer statistischer Größen. Auch hier wird der Abschluss die Verifikation der Verfahren auf Benchmarks und industriellen Problemstellungen in Kap. 11 darstellen. Schließlich fassen wir zusammen und geben einen Ausblick in die Zukunft.

Teil I

Reinforcement-Learning und statistische Lerntheorie

Kapitel 2

Reinforcement-Learning und dynamische Programmierung

Wesentliches Ziel dieser Arbeit ist es, Reinforcement-Learning und Erkenntnisse der statistischen Lerntheorie so miteinander zu verbinden, dass die Informationseffizienz bereits bekannter RL-Verfahren gesteigert werden kann. Zur Motivation werden in diesem Teil beide Themen, Reinforcement-Learning und statistische Lerntheorie, im Überblick vorgestellt.

Wir beschreiben zunächst das allgemeine Optimalregelungsproblem sowie Markov-Entscheidungsprozesse und erläutern, wie dynamische Programmierung zu deren Lösung zur Anwendung kommt. Schließlich erweitern wir unsere Betrachtungen um den Lösungsansatz Reinforcement-Learning, der in einem breiteren Feld eingesetzt werden kann.

Im Anschluss führen wir in Kap. 3 in die statistische Lerntheorie ein und stellen die Prinzipien der strukturellen Risikominimierung (Structural Risk Minimisation) und des Bayes'schen Lernens (Bayesian Learning) gegenüber, auf die wir später aufbauen werden. Dazu werden wir auf die grundlegenden Thesen und Modelle beider Paradigmen eingehen und aufzeigen, worin ihre Gemeinsamkeiten und Unterschiede bestehen.

Schließlich werden zur Skizzierung des Standes der Technik in Kap. 4 einige RL-Verfahren erläutert, bei denen Dateneffizienz im Vordergrund steht oder die durch Dateneffizienz hätten motiviert sein können. Diese Arbeiten dienen als Grundlage für die Beiträge der vorliegenden Arbeit.

2.1 Markov-Entscheidungsprozesse

Ausgangspunkt jedes Optimalregelungsproblems ist ein System, etwa eine technische Anlage, das mittels äußerer Einflüsse gesteuert oder geregelt werden kann. In zeitdiskreten Systemen, die wir hier untersuchen, geht man davon aus, dass sich das System zu unterschiedlichen Zeitpunkten jeweils in einem Zustand $s \in S$ aus einer Menge von möglichen Zuständen befindet und jeweils eine Menge von möglichen Aktionen $a \in A$ zur Auswahl hat. Nach Anwendung der Aktion erfolgt eine Transition, die gegebenenfalls in einen anderen Zustand führt. Im Kontext von RL wird zumeist der Begriff des Agenten verwendet, der sich innerhalb des Zustandsraumes bewegt. Auf diese Weise kann jedes Regelungsproblem auf das Verhalten eines Agenten innerhalb eines Konfigurationsraumes zurückgeführt werden.

Zustandsübergänge erfolgen gemäß einer Transitionswahrscheinlichkeit P_T. Demnach ist zum Zeitpunkt der Auswahl der Aktion der Folgezustand nicht zwangsläufig bekannt, vielmehr unterliegt er einer Stochastizität. Damit nun überhaupt ein Optimierungsproblem vorliegt, ist eine Zielfunktion bzw. ein Kriterium für Optimalität erforderlich. Ihre Grundlage wird durch das Vereinnahmen oder Sammeln (collect) von Belohnungen (Rewards) während der Transitionen durch den Agenten geboten. Prinzipiell besteht also das Ziel, möglichst hohe Rewards zu vereinnahmen.

Soweit die Übergangswahrscheinlichkeit nach s' nur von aktuellem Zustand und gewählter Aktion, also nicht von der gesamten vergangenen Trajektorie abhängt, bezeichnet man $M = (S, A, P, R)$ als Markov-Entscheidungsprozess. Es gilt dann insbesondere

$$P(s'|s,a) \quad = \quad P(s'|s,a,\ldots,s_1,s_0), \tag{2.1}$$

wobei P stets allgemein für Wahrscheinlichkeit steht. Diese als Markov-Eigenschaft bezeichnete Voraussetzung ist entscheidend für das effiziente Lösen von Optimalsteuerungs- und -regelungsproblemen, da alle möglichen auftretenden zukünftigen Entwicklungen des Prozesses und deren Wahrscheinlichkeit ausschließlich aus der Kenntnis des aktuellen Zustandes abgeleitet werden können und vergangene Zustände keinen direkten Einfluss mehr nehmen. Autonome Prozesse, die die Markov-Eigenschaft aufweisen, jedoch über keinen Aktionsraum verfügen, werden als Markov-Prozesse bezeichnet. Dann gilt entsprechend $P(s'|s) = P(s'|s,\ldots,s_1,s_0)$. Zustandsübergänge erfolgen gemäß einer Transitionswahrscheinlichkeit $P_T : S \times A \times S \to [0,1]$, abhängig von Zustand, Aktion und Folgezustand. Die Rewards $R(s,a,s')$ sind im Allgemeinen stochastisch, abhängig von Zustand, gewählter Aktion und erreichtem Folgezustand. Sie unterliegen einer Reward-Wahrscheinlichkeit mit Dichtefunktion $P_R : S \times A \times S \times \mathbb{R} \to \mathbb{R}$, die zu dem erwarteten Reward

$$R(s,a,s') \quad = \quad \int_{-\infty}^{\infty} r P_R(s,a,s',r)dr, \quad s,s' \in S, a \in A \tag{2.2}$$

führt. In vielen praktischen, vor allem technischen Anwendungen besteht jedoch lediglich eine deterministische Abhängigkeit ausschließlich vom Folgezustand. Das Maß an Belohnung ergibt sich dann unmittelbar aus dem erreichten Zustand.

Unter allen oben genannten Voraussetzungen besteht das Ziel nun darin, eine Handlungsstrategie (Policy) $\pi : S \to A$ zu finden, die sich optimal verhält in dem Sinne, dass für alle Zustände s aus S, der Wert von

$$V^{\pi}(s) \quad = \quad \mathbf{E}_{\mathbf{s}}^{\pi} \sum_{t=0}^{\infty} \gamma^t R\left(s^{(t)}, \pi\left(s^{(t)}\right), s^{(t+1)}\right) \tag{2.3}$$

mit $\mathbf{s} = (s', s'', \ldots)$ maximiert wird. Offensichtlich werden nicht nur einzelne Belohnungen während der nächsten Transitionen berücksichtigt, sondern die Summe aller noch erwarteten Rewards. Der Skalar $0 < \gamma < 1$ wird als Diskontierungsfaktor bezeichnet und bewirkt eine geometrische Abschwächung weiter in der Zukunft liegender Rewards. Dies hat den Effekt, dass der Wert $V^{\pi}(s)$ beschränkt ist für alle s, wenn $R(s,a,s')$ beschränkt ist. Damit wird die numerische Handhabbarkeit gewährleistet und das sinnvolle Ziel, hohe Rewards möglichst früh zu vereinnahmen, forciert. Neben der diskontierten Summe der Rewards sind in der Literatur eine Reihe alternativer Kriterien beschrieben worden [107, 13]. Wir werden uns in dieser Arbeit jedoch auf diese Problemstellung konzentrieren und bezeichnen die diskontierte Summe zukünftiger Rewards als Gesamtertrag.

Die Funktionen V^{π} und π werden als Bewertungsfunktion (Value-Function) und Policy bezeichnet und sind von zentraler Bedeutung. Der Wert $V^{\pi}(s)$ ist identisch mit dem erwarteten Gesamtertrag ab Zustand s unter Anwendung der Policy π. Mindestens eine Policy π^* wird als optimal bezeichnet, falls

$$\forall \pi : \forall s \in S : V^{\pi^*}(s) \quad \geq \quad V^{\pi}(s). \tag{2.4}$$

Interessanterweise entsteht hier kein Trade-Off zwischen verschiedenen Zuständen, wie etwa in [78] bewiesen wird, aber auch intuitiv einleuchtet. Eine optimale Policy ist also stets global optimal. Die Menge aller möglichen Bewertungsfunktionen V^{π} bzw. Policies unterliegt jedoch nur einer Halbordnung mit der Ordnungsrelation definiert durch Gl. 2.4. Hervorzuheben ist, dass erst das Konzept der MDP überhaupt die Definition von Bewertungsfunktion und Policy ermöglicht, die nur vom aktuellen Zustand abhängen. Anderenfalls könnten sie nicht die vollständige benötigte Information enthalten.

Die Policy kann alternativ auch als Funktion $\pi : S \times A \rightarrow [0, 1]$ definiert sein, wobei $\pi(s, a)$ die Wahrscheinlichkeit beschreibt, mit der Aktion a im Zustand s ausgeführt wird. Obwohl stets eine deterministische Policy optimal ist [78], gibt es zahlreiche Anwendungen, bei denen stochastische Policies von Bedeutung sind, etwa bei anderen Optimalitätskriterien. In dieser Arbeit werden wir im Zusammenhang mit der Kernel-Rewards-Regression (siehe Abschn. 6.4) und dem Konzept der Sicherheitsoptimalität (siehe Abschn. 9.5) darauf zurückkommen.

2.2 Dynamische Programmierung zur Lösung von Optimalsteuerungs- und -regelungsproblemen

Richard Bellman gilt als Pionier der dynamischen Programmierung (DP) und hat den Begriff geprägt [11]. DP ist auf eine Vielzahl von Optimierungsproblemen anwendbar, Bellman hat die Technik zunächst jedoch zur Optimierung von Regelungsprozessen entwickelt.

Das Grundprinzip ähnelt dem Divide-and-Conquer-Verfahren, ein großes Problem in mehrere Teilprobleme zu untergliedern, um eine rekursive Lösung zu bestimmen. Dabei werden Teillösungen gespeichert und wiederverwendet, so dass eine im Vorhinein bekannte, überschaubare Zahl von Berechnungsschritten notwendig ist. Typische Anwendungen dynamischer Programmierung sind Graphen-Algorithmen, etwa zur Bestimmung kürzester Wege, wie der Dijkstra- und der Floyd-Warshall-Algorithmus. Ein bekanntes Beispiel ist darüber hinaus der Viterbi-Algorithmus zur Bestimmung des kürzesten umfassenden Strings, der auch in lerntheoretischen Problemstellungen zum Einsatz kommt. Einige einführende Ausführungen zur dynamischen Programmierung finden sich bei Cormen et al. [16]. Die Anwendung dieses Verfahrens ist grundsätzlich immer dann möglich, wenn Bellmans Optimalitätsprinzip wirksam ist. Er schreibt selbst, indem er sich auf Optimalsteuerungs- und -regelungsprobleme bezieht:

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [11].

Eine optimale Policy verfügt also über die Eigenschaft, dass, unabhängig davon, was der initiale Zustand und die initialen Entscheidungen sind, die verbleibenden Entscheidungen vom Zustand ausgehend, der aus den ersten Entscheidungen folgt, ebenfalls eine optimale Policy darstellen. Eine formale Definition von Optimierungsproblemen, die sich mit DP lösen lassen, für die also Bellmans Optimalitätsprinzip gilt, findet sich etwa in [35]. Die entscheidenden Eigenschaften sind sowohl die rekursive Darstellbarkeit, also die Aufteilbarkeit in Teilprobleme, die Selbstähnlichkeit der Lösung des Gesamtproblems und der der Teilprobleme und die Wiederverwendbarkeit der Lösungen von Teilproblemen. Dabei muss sich das Gesamtproblem wegen der polynomiellen Anzahl an elementaren Teilproblemen in polynomieller Zeit lösen lassen.

2.2.1 Die Bellman-Gleichung

Mit Hilfe der Bewertungsfunktion sind nun Optimalregelungsprobleme in MDP mit DP lösbar. Denn Gl. 2.3 führt zu der erforderlichen Selbstähnlichkeit. Offensichtlich gilt

$$
\begin{aligned}
V^{\pi}(s) &= \mathbf{E}^{\pi}_{s'}(R(s, \pi(s), s') + \gamma V^{\pi}(s')) && (2.5)\\
&= \mathbf{E}^{\pi}_{s'}(R(s, \pi(s), s')) + \gamma \mathbf{E}^{\pi}_{s',s''}(R(s', \pi(s'), s'') + \gamma V^{\pi}(s'')) && (2.6)\\
&= \mathbf{E}^{\pi}_{s',s''}\big(R(s, \pi(s), s') + \gamma R(s', \pi(s'), s'') + \gamma^2 V^{\pi}(s'')\big) && (2.7)\\
&= \dots \\
&= \mathbf{E}^{\pi}_{\mathbf{s}} \sum_{t=0}^{\infty} \gamma^t R\left(s^{(t)}, \pi\left(s^{(t)}\right), s^{(t+1)}\right). && (2.8)
\end{aligned}
$$

Gl. 2.5 wird als Bellman-Gleichung bezeichnet und ermöglicht die Anwendung von DP zur Bestimmung der Bewertungsfunktion V^π für π unter der Voraussetzung, dass P und R vollständig bekannt sind. Die alternative Schreibweise

$$V^\pi(s) = \mathbf{E}^\pi_{s'}(R(s,\pi(s),s') + \gamma V^\pi(s')) \quad (2.9)$$

$$= \sum_{s' \in S} P(s'|s,\pi(s)) \left(R(s,\pi(s),s') + \gamma V^\pi(s')\right) \quad (2.10)$$

führt unmittelbar zur algorithmischen Lösung (siehe Alg. 1). Bezogen auf die Bellman-

Algorithmus 1 Dynamische Programmierung für Policy-Evaluation

Bedingungen: gegebene Beobachtungen P und R eines diskreten MDP, ein Skalar $0 < \gamma < 1$ sowie eine deterministische Policy π

Zusicherungen: bestimmt die Bewertungsfunktion V^π

setze $\forall s \in S : V^0(s) = 0$

setze $t = 0$

solange die gewünschte Genauigkeit nicht erreicht ist **führe aus**

setze $t = t + 1$

setze $\forall s \in S : V^t(s) = \sum_{s' \in S} P(s'|s,\pi(s)) \left(R(s,\pi(s),s') + \gamma V^{t-1}(s')\right)$

ende solange

gebe V^t zurück

Gleichung (Gl. 2.5) repräsentiert $V^t(s)$ genau die Summe für $V^\pi(s)$, jedoch abgeschnitten nach dem $t$ten Summanden. Durch das Diskontieren konvergiert dieser Prozess zwangsläufig, in dem Sinne, dass $|V^t(s) - V^{t-1}(s)| \to 0$ für $t \to \infty$. Das Bestimmen der Lösung der Bellman-Gleichung wird als Policy-Evaluation bezeichnet.

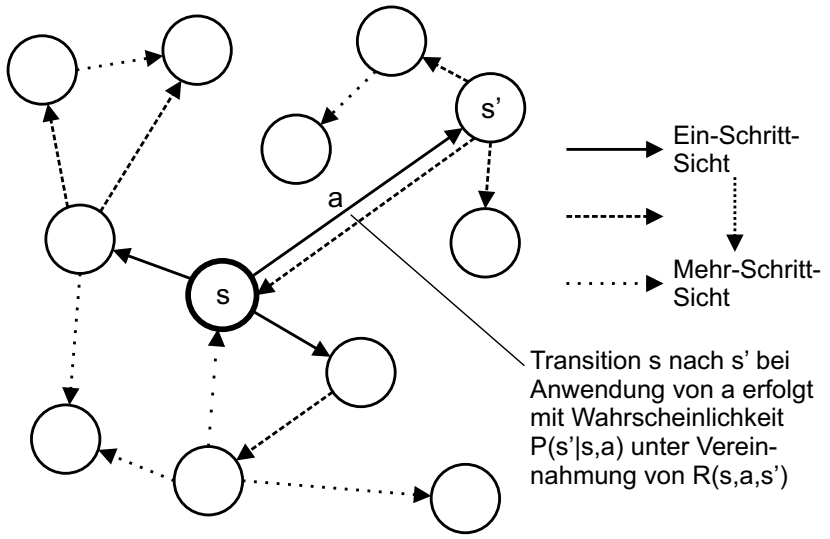


Abbildung 2.1: Illustration der dynamischen Programmierung für Markov-Entscheidungsprozesse. Mit jedem Iterationsschritt wird die Umgebung um eine Schale weiter erkundet, schließlich beinhaltet die Bewertungsfunktion den Gesamtertrag, der auf dem gesamten MDP beruht.

2.2.2 Die Bellman-Optimalitäts-Gleichung

Mit der Policy-Evaluation wird allerdings das Optimalregelungsproblem nicht gelöst, schließlich war die Policy bereits bekannt und Alg. 1 bestimmt lediglich die dazugehörige Bewertungsfunktion.

Zur Ermittlung der optimalen Bewertungsfunktion V^* und der dazugehörigen Policy π^* muss Policy-Evaluation mit Policy-Improvement verbunden werden. Dazu wird eine übergeordnete Schleife eingeführt, in der die temporär optimale Policy

$$\forall s \in S : \pi(s) \quad = \quad \arg\max_a \sum_{s' \in S} P(s'|s,a) \left(R(s,a,s') + \gamma V(s')\right) \tag{2.11}$$

bestimmt und für das folgende V^t betrachtet wird. Dieses Vorgehen wird i.e.S. als Policy-Iteration bezeichnet [107] und führt zur Konvergenz gegen die optimale Policy π^*, wie leicht zu zeigen ist [107].

Policy-Iteration in dieser Form ist aufwändig einerseits der doppelten Verschachtelung und andererseits der aufwändigen Aktualisierung der Policy wegen. Aus diesem Grund findet, neben der Bewertungsfunktion, die sogenannte Qualitätsfunktion (Q-Funktion) Anwendung, die von Zustand und gewählter Aktion abhängt. Analog zur Bewertungsfunktion ergibt sich die Bellman-Gleichung

$$\begin{aligned} Q^\pi(s,a) &= \mathbf{E}_{s'}(R\left(s,a,s'\right) + \gamma Q^\pi(s', \pi(s'))) && (2.12)\\ &= \mathbf{E}_{s'}(R\left(s,a,s'\right) + \gamma V^\pi(s')), && (2.13) \end{aligned}$$

die im Unterschied zu Gl. 2.5 auch andere unmittelbare Aktionen vorsieht als die, die durch π vorgegeben werden. Die erste Transition in Zustand s kann also mittels Aktion a erfolgen. Im Anschluss daran folgt der Agent π. Die Q-Funktion beinhaltet mehr Informationen als die Bewertungsfunktion, die man sich zu Nutze macht. Denn offensichtlich kann Gl. 2.11 direkt mit Hilfe der Q-Funktion berechnet werden. Darüber hinaus erlaubt die Q-Funktion die Formulierung einer Bellman-Optimalitäts-Gleichung

$$\begin{aligned} Q^*(s,a) &= \mathbf{E}_{s'}\Big(R\left(s,a,s'\right) + \gamma \max_{a'} Q^*(s',a')\Big) && (2.14)\\ &= \mathbf{E}_{s'}(R\left(s,a,s'\right) + \gamma V^*(s')), && (2.15) \end{aligned}$$

die unabhängig von π ist, deren Lösung jedoch zur optimalen Policy π^* führt, indem

$$\pi^*(s) \quad = \quad \arg\max_a Q^*(s,a) \tag{2.16}$$

gewählt wird.

Der durch die Bellman-Gleichung realisierte Operator T^π mit

$$(T^\pi Q)(s,a) \quad = \quad \mathbf{E}_{s'}(R\left(s,a,s'\right) + \gamma Q(s', \pi(s'))) \tag{2.17}$$

wird auch als Bellman-Operator bezeichnet. I.e.S. ist der Bellman-Operator T für die Bellman-Optimalitäts-Gleichung als

$$(TQ)(s,a) \quad = \quad \mathbf{E}_{s'}\Big(R\left(s,a,s'\right) + \gamma \max_{a'} Q(s',a')\Big) \tag{2.18}$$

definiert. Ferner definieren wir

$$\begin{aligned} ((T')^\pi Q)(s,a) &= \mathbf{std}_{s'}(R\left(s,a,s'\right) + \gamma Q(s', \pi(s'))) && (2.19)\\ (T'Q)(s,a) &= \mathbf{std}_{s'}\Big(R\left(s,a,s'\right) + \gamma \max_{a'} Q(s',a')\Big) && (2.20) \end{aligned}$$

als die Standardabweichung der diskontierten Summe zukünftiger Rewards.

Dass die Lösung von Gl. 2.14 tatsächlich die optimale Q-Funktion ist, folgt per indirektem Beweis. Man nimmt an, dass Q^* nicht optimal ist und schließt daraus, dass es mindestens einen Zustand s geben müsse, für den ein a existiere, so dass $Q^*(s,\pi^*(s)) < \max_{a'} Q^*(s,a') = Q^*(s,a)$. Dann kann Q^* aber bereits nicht Gl. 2.14 erfüllen, wenn gleichzeitig Gl. 2.12 erfüllt ist.

Analog zu Alg. 1 beschreibt Alg. 2 das Vorgehen der dynamischen Programmierung für Policy-Iteration, die hier i.e.S. als Value-Iteration bezeichnet wird. Der algorithmische Aufwand ist nicht wesentlich größer, obwohl ein dem Anschein nach schwierigeres Problem gelöst wird. Die Q-Funktion und die Bellman-Optimalitäts-Gleichung sind daher bahnbrechende Konzepte. Policy-Evaluation und Policy-Iteration durch Value-Iteration werden auch unter dem Begriff Bellman-Iteration zusammengefasst.

Algorithmus 2 Dynamische Programmierung für Policy-Iteration

Bedingungen: gegebene Beobachtungen P und R eines diskreten MDP und ein Skalar $0 < \gamma < 1$

Zusicherungen: bestimmt die optimale Q-Funktion Q^* und die optimale Policy π^*

setze $\forall s \in S, a \in A : Q^0(s,a) = 0$
setze $t = 0$
solange die gewünschte Genauigkeit nicht erreicht ist **führe aus**
  setze $t = t + 1$
  setze $\forall s \in S : V^{t-1}(s) = \max_{a'} Q^{t-1}(s,a')$
  setze $\forall s \in S, a \in A : Q^t(s,a) = \sum_{s \in S} P(s'|s,a) \left(R(s,a,s') + \gamma V^{t-1}(s') \right)$
ende solange
setze $\forall s \in S : \pi(s) = \arg\max_{a'} Q^t(s,a')$
gebe Q^t und π zurück

2.2.3 Partiell beobachtbare, nicht-stationäre und Markov-Entscheidungsprozesse höherer Ordnung

In vielen praktischen Anwendungen kann die Markov-Eigenschaft nicht garantiert werden. Dies wird zur Bewältigung industrieller Probleme noch von Bedeutung sein (siehe Abschn. 8.2), da dort die Messungen i.d.R. nicht ausreichen, um das System vollständig zu beschreiben. Bei partiell beobachtbaren (partially observable) Markov-Entscheidungsprozessen (POMDP) wird zwischen Zustands- und Beobachtungsraum U unterschieden. Eine Funktion $f : S \to U$ weist jedem Zustand eindeutig eine Beobachtung zu, die dem Agenten, anstatt des Zustandes selbst, zur Verfügung gestellt wird. Im Allgemeinen ist die Markov-Eigenschaft für U also nicht mehr erfüllt.

Eine den POMDP untergeordnete Problemklasse sind die MDP höherer Ordnung. Ein MDP mit Zustandsraum S und Aktionsraum A hat Ordnung n, falls

$$P\left(s^{(n)} \left| s^{(n-1)}, a^{(n-1)}, \ldots, s \right.\right) = P\left(s^{(n)} \left| s^{(n-1)}, a^{(n-1)}, \ldots, s, \ldots, s_1, s_0 \right.\right). \quad (2.21)$$

Mit anderen Worten, nicht ausschließlich der aktuelle Zustand ist ausschlaggebend für die Kenntnis aller möglichen zukünftigen Entwicklungen des Prozesses und deren Wahrscheinlichkeiten, sondern die letzten n Zustände.

In nicht-stationären MDP gibt es keine festen Transitionswahrscheinlichkeiten P_T, diese können sich vielmehr während des Prozesses verändern. Hier ist die Zeitinvarianz, die in der Bellman-Gleichung und der Bellman-Optimalitäts-Gleichung inhärent angenommen wird, verletzt.

Nicht-stationäre MDP lassen sich auf POMDP zurückführen, wenn die Veränderung von P_T selbst einem MDP unterliegt. Obwohl POMDP eine mächtigere Problemklasse als MDP höherer Ordnung darstellen, sind konkrete Problemstellungen in der Praxis oft identisch. Wenn eine beobachtbare mit einer nicht beobachtbaren Größe physikalisch zusammenhängt, können aus der Historie der beobachtbaren Größe oftmals Rückschlüsse auf die nicht beobachtbare Größe gezogen werden. So kann bei gleichförmiger Bewegung aus zwei Positionen etwa die Geschwindigkeit abgeleitet werden. Entsprechendes gilt für die Ableitbarkeit der Beschleunigung aus drei Positionsbeobachtungen.

2.3 Klassische und modell-prediktive Regelung

Reinforcement-Learning, das auf Markov-Entscheidungsprozesse aufbaut, ist, wie wir in den Abschn. 2.4 und 2.5 erläutern, ein lerntheoretischer Ansatz zur Lösung von Optimalregelungsproblemen. Derartige Probleme werden in der Regelungstechnik bereits seit Mitte des 20. Jahrhunderts bearbeitet. Dabei wird das Reglerdesign nicht datengetrieben, durch Lernen, sondern konstruktiv vollzogen. Man geht dazu davon aus, dass für die Dynamik des

Systems ein Modell zur Verfügung steht, das etwa als Zustandsraum-Modell oder als Transferfunktion beschrieben ist und der Realität entspricht oder dieser zumindest möglichst nahe kommt.

Anstelle eine Reward-Funktion zu maximieren, wird dann das Ziel angestrebt, den Ist-Wert einer Messgröße möglichst dicht an ihren Sollwert heranzuführen (Sollwertfolge). Die Mächtigkeit der Aufgabenstellung kann dadurch erhöht werden, dass der Sollwert dem Wert einer Funktion der Zeit entspricht (Trajektorienfolge), was dem Konzept der Reward-Funktion bereits nahe kommt. Darüber hinaus wird der Übergang zwischen Zuständen stets als deterministisch angenommen, wohingegen Markov-Entscheidungsprozesse im Allgemeinen stochastisch sind.

Ein Beispiel für einen wichtigen Regler ist der sogenannte PID-Regler [23] (Proportional-Integral-Differenzial-Regler), der eine Stellgröße $u(t)$ an Hand des Fehlersignals $e(t)$ gemäß

$$u(t) = K_D \frac{de(t)}{dt} + K_P e(t) + K_I \int_0^t e(t')dt' \tag{2.22}$$

verändert. Für eine analytische Optimierung der Parameter K_D (Differenzialglied), K_P (Proportionalglied) und K_I (Integralglied) ist es erforderlich, dass das Modell des Systems linearisiert wird. Weicht das System dann stark von seinem Modell ab, so kann man an verschiedenen Arbeitspunkten auf unterschiedliche Linearisierungen zurückgreifen und, je nach Zustand, verschiedene Regler oder Linearkombinationen von Reglern anwenden.

Liegt das System (bzw. das Modell des Systems) tatsächlich als LZI-System (lineares zeitinvariantes dynamisches System) vor, dann kann die optimale Wahl der drei Parameter schließlich durch analytische Methoden wie dem Wurzelortskurven- oder dem Frequenzlinienverfahren erfolgen. Anderenfalls und grundsätzlich ist auch eine systematische Suche möglich.

Die mächtigere modell-prediktive Regelung [56, 30] stellt weniger Anforderungen an die Aufgabenstellung. So muss weder ein lineares noch überhaupt ein präzises Modell über die Realität zur Verfügung stehen. Stattdessen kann ein, gegebenenfalls nicht-lineares Modell auch adaptiert werden, das dann i.d.R. in analytischer Form vorliegt und zur Vorhersage zukünftiger Zustände verwendet wird. Die Stellgrößen werden schließlich mit direkten Optimierungsverfahren bzgl. bestimmter Kriterien, etwa der Sollwertfolge, optimiert. Diese verwenden dabei die Zustands-Prognose.

Modell-prediktive Regelung verfügt daher gegenüber der klassischen Regelung über erhebliche Vorteile. So stehen sowohl der Modellausgestaltung als auch der anwendbaren Optimierungskriterien kaum Einschränkungen im Wege. Darüber hinaus kann die Methode physikalisch-technische Beschränkungen des Systems explizit berücksichtigen. Modell-prediktive Regelung ist also eine ernstzunehmende Alternative zum Reinforcement-Learning und hat mit modellbasiertem Reinforcement-Learning (siehe Abschn. 2.4.3) viele Gemeinsamkeiten.

2.4 Reinforcement-Learning als lerntheoretischer Ansatz

Dynamische Programmierung liefert zwar nachweislich die optimale Policy für einen gegebenen MDP mit geringem Aufwand, ist jedoch eingeschränkt auf die Fälle, für die P und R vollständig bekannt sind und von endlichen Zustandsräumen ausgegangen werden kann. Wenn eine von beiden Bedingungen nicht erfüllt ist, kann Reinforcement-Learning zum Einsatz kommen.

Demnach befassen wir uns mit folgender Situation. Der Agent befindet sich in einer Umgebung, die aus bekanntem Zustands- und Aktionsraum besteht und in der zulässige Aktionen ausgeführt werden können. Ferner sei angenommen, dass die Zustände die Markov-Eigenschaft aufweisen. Bei jeder Transition vereinnahmt der Agent einen Reward. Die Auf-

gabe besteht darin, den Agenten optimal innerhalb der Umgebung zu bewegen. Da er zu Beginn über keine detaillierten Kenntnisse über den zu Grunde liegenden MDP verfügt, ist er darauf angewiesen, im Verlaufe seiner Bewegung in der Umgebung sein Verhalten zu erlernen. Dabei wird eine Qualität des RL erkennbar, die in den meisten anderen Machine-Learning-Bereichen nicht berücksichtigt werden kann, der Agent kann aktiv beeinflussen, welche Transitionen er ausführt, um Erkenntnisse über den MDP zu gewinnen.

2.4.1 Das Explorations-Ausbeutungs-Dilemma

Typischerweise besteht ein Trade-Off zwischen der Entscheidung, neue Informationen zu sammeln und sich bereits möglichst gut, also gesamtertrags-maximierend zu verhalten. Diese Problematik wird als Explorations-Ausbeutungs-Dilemma (Exploration-Exploitation-Dilemma) bezeichnet [107]. Um diesem Problem zu begegnen, wird dem Agenten ein Explorationsmechanismus zur Seite gestellt.

Der einfachste ist die Random-Exploration, bei der strikt zwischen Explorations- und Ausbeutungsphase unterschieden wird. Zunächst bewegt sich der Agent zufällig in der Umgebung und lernt die Policy. Dann beginnt er erst, sich produktiv zu verhalten. Bei der sogenannten ε-greedy-Exploration wird ausbeutendes und zufälliges Verhalten kombiniert, mit Wahrscheinlichkeit $P = \varepsilon$ wird eine zufällige Aktion ausgewählt, sonst diejenige, die der aktuell angenommenen optimalen Policy entspricht. Bei der Boltzmann-Exploration ist die Verwendung einer Q-Funktion Voraussetzung. Hier verhält sich der Agent stochastisch, mit Wahrscheinlichkeit

$$P \quad = \quad \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in A} e^{\frac{Q(s,a')}{\tau}}} \tag{2.23}$$

wird Aktion a ausgewählt. Die sogenannte Temperatur τ ist ein Maß für die Stochastizität. Je größer τ, desto zufälliger ist die Exploration. Die Extrema sind Random-Exploration (für $\tau \to \infty$) und Ausbeutung (für $\tau \to 0$). Es existieren eine ganze Reihe weiterer, wesentlich komplexerer Explorationstechniken, bei denen etwa Informationsmaße einbezogen werden (z.B. [5, 1, 38, 39]). Dies führt dazu, dass die Exploration insgesamt zielgerichteter verläuft.

2.4.2 On-Policy- und Off-Policy-Learning

Damit nun parallel zur Exploration der Agent eine optimale Policy erlernen kann, muss ihm ein Online-Lernverfahren zur Seite gestellt werden. Man unterscheidet dabei zwischen den On-Policy- und Off-Policy-Verfahren. On-Policy-Verfahren können keine wesentlich bessere Policy hervorbringen als die, die der Agent im besten Fall während der Exploration benutzen würde. Tatsächlich ist zwar die greedy Policy bzgl. einer Q-Funktion, die einer ε-greedy Policy entspricht, stets mindestens so performant, wie sich direkt aus der Definition der Q-Funktion ergibt [107], jedoch kann sie im Allgemeinen nicht optimal sein. Dies hat zur Folge, dass, um eine optimale Policy erreichen zu können, die Explorationskomponente nach und nach zurückgeschraubt werden muss, bis nur noch ausgebeutet wird.

Der SARSA-Algorithmus [87] (SARSA entspricht State-Action-Reward-State-Action) ist ein On-Policy-Verfahren für endliche Zustands- und Aktionsräume und gehört zur Klasse der sogenannten Temporal-Difference-Learning-Verfahren. Dabei wird nach jeder Transition die Q-Funktion so angepasst,

$$Q_{\text{neu}}(s,a) \quad = \quad Q(s,a) + \alpha \left(r + \gamma Q(s',a') - Q(s,a) \right), \tag{2.24}$$

dass die temporale Differenz dem entsprechenden Q-Wert zugeschlagen wird, gewichtet mit einer Lernrate α. Dabei bezeichnen s' und $a' = \pi(s')$ Folgezustand und Folgeaktion sowie r den Reward. Man sieht leicht, dass, falls Q die Q-Funktion der aktuellen Policy ist, die Differenz $r + \gamma Q(s',a') - Q(s,a) = 0$ im Mittel verschwinden muss und sich die Q-Funktion daher nicht mehr verändert.

Obwohl chronologisch Q-Learning vor SARSA entwickelt worden ist, gilt es daher als großer Fortschritt, diese Abhängigkeit von der Exploration überwunden zu haben. Der Q-Learning-Algorithmus [119, 120] ist ein weit verbreitetes Off-Policy-Verfahren. Während SARSA in Analogie zur Bellman-Gleichung arbeitet, bezieht sich Q-Learning direkt auf die Bellman-Optimalitäts-Gleichung und folgt der Lernregel

$$Q_{\text{neu}}(s,a) \quad = \quad Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right). \tag{2.25}$$

Dies hat den entscheidenden Vorteil, dass die temporale Differenz $r + \gamma \max_{a'} Q(s',a') - Q(s,a)$ im Mittel nur dann verschwindet, wenn Q die Q-Funktion der optimalen Policy ist. Der Agent muss sich während der Explorationsphase zu keinem Zeitpunkt optimal verhalten und kann dennoch die optimale Policy bestimmen.

In beiden Verfahren kann Konvergenz garantiert werden, falls die Lernschrittweite sukzessive reduziert wird und

$$\sum_{t=1}^{\infty} \alpha_t \quad = \quad \infty \tag{2.26}$$

$$\sum_{t=1}^{\infty} \alpha_t^2 \quad < \quad \infty \tag{2.27}$$

gilt [107].

2.4.3 Modellfreies und modellbasiertes Reinforcement-Learning

Q-Learning und SARSA sind modellfreie RL-Verfahren, weil sie kein explizites Modell der Transitionswahrscheinlichkeiten aufbauen. Sämtliche ihnen zur Verfügung gestellten Informationen werden ausschließlich zur Modifikation der Q-Funktion benutzt. Ihnen stehen die modellbasierten RL-Verfahren gegenüber.

Modellbasierte RL-Verfahren können mit jeder Transition ein internes Modell der Übergangswahrscheinlichkeiten anpassen. Im einfachsten Fall wird die relative Häufigkeit der Beobachtungen benutzt. Wurden also im Laufe der Exploration $n_{s,a}$ Transitionen von s ausgehend mit Aktion a vollzogen und gelangten davon $n_{s'|s,a}$ in den Zustand s', dann wird $P(s'|s,a) = n_{s'|s,a}/n_{s,a}$ und analog der Reward $R(s,a,s')$ gemäß dem durchschnittlich an dieser Transition beobachteten Reward gewählt. Der Dyna-Q Algorithmus [106] kombiniert nun Q-Learning mit dem Aufbau eines Modells, indem nach jedem Q-Learning-Lernschritt N Simulationsschritte nach dem bis dahin geschätzten Modell ausgeführt werden, jeweils einschließlich eines Q-Learning-Lernschrittes. Diese Technik führt zu schnellerer Konvergenz.

Natürlich kann nach jeder Transition auch alternativ das Modell aktualisiert werden und DP zur Anwendung kommen (siehe Alg. 2), unter der Annahme, dass das Modell der Wirklichkeit entspricht. Der Prioritised-Sweeping-Algorithmus [65, 71] (siehe Alg. 3) ist eine effiziente Methode, um, je nach konkreter Ausprägung des Algorithmus, Q-Learning oder dynamische Programmierung zu benutzen, um die für das Modell optimale Lösung zu bestimmen.

2.5 Grundlegende Lösungsverfahren für komplexe Probleme

Die oben beschriebenen Verfahren sind ausschließlich für MDP mit überschaubar großem Zustandsraum geeignet. In kontinuierlichen, hochdimensionalen Zustandsräumen, wie sie in der Praxis häufig vorkommen, sind weder DP noch SARSA oder Q-Learning anwendbar.

Dennoch werden in den meisten kontinuierlichen RL-Verfahren die Konzepte der Bewertungsfunktion und der Q-Funktion nach wie vor verwendet. Dabei muss die Q-Funktion $Q \in H_Q$ aus einem geeigneten Funktionsraum gewählt werden, da sie nicht mehr als Tabelle

Algorithmus 3 Prioritised-Sweeping

Bedingungen: gegeben ein diskreter MDP sowie Skalare $0 < \gamma < 1$ und μ
Zusicherungen: bestimmt die optimale Q-Funktion Q^* und die optimale Policy π^*

```
setze ∀i, j : Q(s_i, a_j) = 0
initialisiere eine Prioritised-Queue PQ = leer
solange Explorations-Phase fortgesetzt wird führe aus
  beobachte Transition (s, a, r, s')
  aktualisiere Modell mit (s, a, r, s')
  setze p = |r + γ max_a' Q(s', a') − Q(s, a)|
  wenn p > μ dann
    füge (s, a) in PQ mit Priorität p ein
  ende wenn
  solange PQ nicht leer ist führe aus
    setze (s, a) = PQ.vorne
    simuliere Modell auf (s, a) und beobachte s', r
    setze Q(s, a) = Q(s, a) + α (r + γ max_a' Q(s', a') − Q(s, a))
    für alle (s̄, ā, r̄), die zu s führen führe aus
      setze p = |r̄ + γ max_a Q(s, a) − Q(s̄, ā)|
      wenn p > μ dann
        füge (s̄, ā) in PQ mit Priorität p ein
      ende wenn
    ende für
  ende solange
ende solange
setze ∀i : π(s_i) = arg max_a' Q(s_i, a')
gebe Q und π zurück
```

darstellbar ist. In Kapitel 3 gehen wir noch auf prinzipielle Fragen zur Wahl von Hypothesenräumen ein.

2.5.1 Monte-Carlo-Methoden

Das Prinzip der Monte-Carlo-Methoden ist konzeptionell einfach und erfolgversprechend. Der Agent startet in einem beliebigen (zufälligen) zulässigen Ausgangszustand, führt eine Transition mit einer zufälligen (oder gemäß einer Explorationsstrategie) Aktion aus und vollzieht im Anschluss N weitere Transitionen mit einer zuvor festgelegten Policy π, etwa der zufällig agierenden Policy. Dies wird mehrere Male wiederholt, so dass ein Datensatz, bestehend aus den Startzuständen s_i, Startaktionen a_i und der diskontierten Summe zukünftiger Rewards

$$D(s_i, a_i) \quad = \quad R(s_i, a_i, s_i') + \sum_{t=1}^{N} \gamma^t R\left(s_i^{(t)}, \pi\left(s_i^{(t)}\right), s_i^{(t+1)}\right) \tag{2.28}$$

aufgebaut wird. Auf diese Weise entsteht ein Regressionsproblem, bei dem die Beobachtungen (s_i, a_i) auf $D(s_i, a_i)$ abgebildet werden müssen. Per Konstruktion beschreibt das Ergebnis Q eine Schätzung der Q-Funktion auf dem MDP für die Policy π. Nun wird π mit der bzgl. Q optimalen Policy ersetzt und der gesamte Vorgang wiederholt, bis keine Veränderung der Policy mehr möglich ist, so dass die optimale Policy erreicht ist.

Dadurch, dass eine Summe aus tatsächlich beobachteten Rewards benutzt wird, ist $D(s_i, a_i)$ ein erwartungstreuer Schätzer für $Q(s_i, a_i)$, zumindest für $N \to \infty$. Dies ist der entscheidende Vorteil der Monte-Carlo-Methoden. Die entscheidenden Nachteile liegen auf der Hand, ist der MDP stark stochastisch, unterliegt die Schätzung $D(s_i, a_i)$ für $Q(s_i, a_i)$ einer hohen Varianz, da nach N Schritten bereits viele zukünftige Entwicklungen möglich

sind. Darüber hinaus ist ein immenser Explorationsaufwand erforderlich, bis eine angemessene Policy bestimmt werden kann. Dadurch, dass nach jedem Monte-Carlo-Schritt sämtliche zuvor gesammelten Daten nicht mehr berücksichtigt werden können, da sie einer anderen Policy zu Grunde liegen, handelt es sich um ein On-Policy-Verfahren und die Technik ist kein Kandidat für informationseffizientes RL.

In [107, 73] sind allerdings Methoden zum Off-Policy-Monte-Carlo-RL beschrieben. Tatsächlich ist es im Rahmen dieses Ansatzes also möglich, mit einer Policy π' zu simulieren und die beobachteten Transitionen zur Bestimmung der Q-Funktion einer von π' verschiedenen Policy π zu verwenden. Dazu kann jedoch nur ein Teil der beobachteten Transitionen benutzt werden, der umso kleiner wird, je stärker sich π von π' unterscheidet. Die prinzipiellen Einschränkungen bleiben also bestehen.

2.5.2 Temporal-Difference-Learning

Temporal-Difference-Learning (TD-Learning) [105, 109, 112] ist ebenfalls ein Off-Policy-Verfahren, wird i.e.S. als die kontinuierliche Variante des Q-Learnings bezeichnet und i.w.S. dem Q-Learning übergeordnet. Der wesentliche Unterschied besteht wieder darin, dass Q eine kontinuierliche Funktion ist und nicht aus einer endlichen Anzahl von Einträgen besteht. Wenn $Q = Q_\theta \in H_Q$ vollständig beschrieben werden kann durch einen Satz von Parametern θ, in denen Q_θ differenzierbar ist, ergibt sich die TD-Lernregel als

$$\theta_{\text{neu}} \quad = \quad \theta + \alpha \left(\hat{Q}(s,a) - Q_\theta(s,a) \right) \frac{\partial}{\partial \theta} Q_\theta(s,a), \tag{2.29}$$

mit $\hat{Q}(s,a)$ als temporären Schätzer für $R(s,a,s') + \gamma V^*(s')$, für den etwa $r + \gamma \max_{a'} Q_\theta(s',a')$ gewählt werden kann. Die Lernregel ergibt sich unmittelbar aus dem Ziel, den lokalen quadratischen Fehler $\left(\hat{Q}(s,a) - Q_\theta(s,a) \right)^2$ zu minimieren und entspricht einem Gradientenabstieg. Falls $\hat{Q}(s,a)$ erwartungstreu ist, kann Konvergenz gegen die optimale Lösung innerhalb von H_Q garantiert werden. Dies ist jedoch im Allgemeinen nicht gewährleistet, da die optimale Q-Funktion $Q^* \notin H_Q$ nicht zwangsläufig im vorgesehenen Hypothesenraum liegen muss.

Eine Variante des TD-Learnings sind die Residual-Gradient-Algorithmen [6], die anstatt des lokalen Fehlers, also der temporalen Differenz, ein globales Fehlermaß minimieren. Dazu wird der Gradient des Fehlers $(r + \gamma \max_{a'} Q_\theta(s',a') - Q_\theta(s,a))^2$ bzgl. θ vollständig berücksichtigt und es ergibt sich die Lernregel

$$\begin{aligned} \theta_{\text{neu}} \quad = \quad & \theta + \alpha \left(r + \gamma \max_{a'} Q_\theta(s',a') - Q_\theta(s,a) \right) \\ & \left(\frac{\partial}{\partial \theta} Q_\theta(s,a) - \gamma \frac{\partial}{\partial \theta} Q_\theta \left(s', \arg\max_{a'} Q_\theta(s',a') \right) \right). \end{aligned} \tag{2.30}$$

Diese Methode führt zum Minimieren des sogenannten Bellman-Residuums, ist aber, wie auch das TD-Learning, nicht erwartungstreu. Dies wird in den Abschn. 4.2 und 5.4 noch im Detail untersucht. Beide Lösungsansätze sind Grundlage für die Rewards-Regression (siehe Kap. 6 und 7).

2.5.3 Policy-Gradient- und Actor-Critic-Verfahren

Policy-Gradient-Methoden [108] unterscheiden sich grundlegend von den bisher vorgestellten Ansätzen. Denn hier werden weder Bewertungsfunktion noch Q-Funktion, sondern direkt die Policy optimiert. Das Prinzip besteht grundsätzlich darin, die Policy in einem Lernschritt so zu modifizieren, dass ein zuvor festgelegtes Performance-Maß V optimiert wird, etwa der Gesamtertrag oder der durchschnittliche Reward. Ist die Policy $\pi = \pi_\psi \in \Pi$ mit Hilfe von Parametern ψ vollständig beschreibbar, dann kann die Performance der Policy mittels

Gradientenaufstieg in Richtung

$$\Delta\psi \quad = \quad \frac{\partial V}{\partial \psi} \tag{2.31}$$

verbessert werden. Sutton et al., die den Begriff Policy-Gradient geprägt haben, gehen dabei von stochastischen Policies aus und haben gezeigt, dass für beide Performance-Maße der Zusammenhang

$$\frac{\partial V}{\partial \psi} \quad = \quad \int_{s\in S} P_\pi(s) \sum_{a\in A} \frac{\partial \pi(s,a)}{\partial \psi} Q^\pi(s,a) ds \tag{2.32}$$

mit $P_\pi(s)$ der stationären Wahrscheinlichkeit des Auftretens von Zustand s bei Anwendung von π, gilt [108]. Dabei ist Q^π die wahre Q-Funktion bzw. ein erwartungstreuer Schätzer. Sutton et al. zeigen ferner, dass Gl. 2.32 auch gilt, falls Q^π durch eine lokal optimale Schätz-Funktion $\hat{Q}^\pi$ der Gestalt, dass zusätzlich

$$\frac{\partial \hat{Q}^\pi(s,a)}{\partial \psi} \quad = \quad \frac{\partial \pi(s,a)}{\partial \psi} \frac{1}{\pi(s,a)} \tag{2.33}$$

gewährleistet werden kann, approximiert wird (Details dazu in [108]). Das heißt, die Policy-Gradient-Schätzung bleibt korrekt, obwohl der Schätzer für Q^π nicht erwartungstreu ist. Weiterentwicklungen dieses Ansatzes sind beschrieben u.a. in [50].

Die eng mit den Policy-Gradient-Methoden verwandten Actor-Critic-Methoden [9, 54, 77] verzahnen Policy und Q-Funktion enger miteinander, indem sie sie sich wechselseitig beeinflussen lassen. So kann etwa eine (deterministische) Policy π_ψ zur Exploration benutzt werden. Währenddessen wird Q jedoch mit On-Policy-TD-Learning optimiert, so dass sich die Lernregel

$$\theta_{\text{neu}} \quad = \quad \theta + \alpha(r + \gamma Q_\theta(s, \pi_\psi(s)) - Q_\theta(s,a)) \frac{\partial}{\partial \theta} Q_\theta(s,a), \tag{2.34}$$

ergibt. Nach jedem TD-Lernschritt wird die Policy in geeigneter Weise angepasst, so dass etwa

$$Q_\theta(s, \pi_\psi(s)) \quad \rightarrow \quad \max. \tag{2.35}$$

Der Begriff Actor-Critic ist abgeleitet von der Verwendung einer expliziten Policy (dem Actor) und einer kritisierenden Q-Funktion (dem Critic). Der Vorteil der Actor-Critic-Verfahren

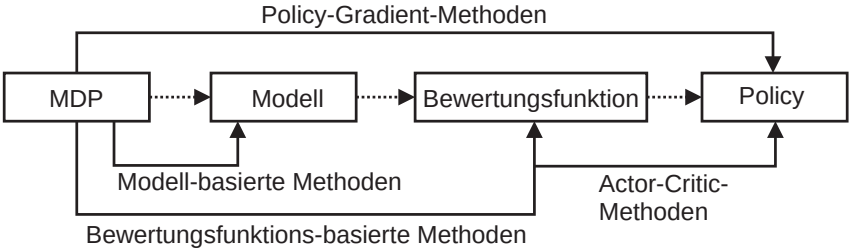


Abbildung 2.2: Prinzip der Adaption verschiedener Klassen von Reinforcement-Learning-Verfahren. Das Lernen einer Policy erfolgt in unterschiedlichen Zwischenschritten. Während Policy-Gradient-Verfahren die Policy aus der Umgebung ableiten, geschieht dies etwa bei Temporal-Difference-Learning über den Zwischenschritt der Bewertungsfunktion. Modellbasierte Verfahren konstruieren zunächst ein Modell des MDP.

besteht vor allem darin, dass problemlos kontinuierliche Aktionen berücksichtigt werden können, da keine Maximumsbestimmung der Q-Funktion mehr erforderlich ist. Außerdem lässt sich die Policy geeignet parametrisieren, falls Vorwissen über ihre Form zur Verfügung steht. In POMDP, also in Umgebungen, deren Zustände nicht über die Markov-Eigenschaft verfügen, lassen sich Actor-Critic- und Policy-Gradient-Methoden oftmals erfolgreicher einsetzen als ausschließlich bewertungsfunktions-basierte Methoden [47].

Kapitel 3

Statistische Lerntheorie

Die statistische Lerntheorie liefert eine theoretische Grundlage für Klassifikation und Regression. In diesem Kapitel soll daher eine Einführung in den zweiten Grundbaustein dieser Arbeit erfolgen. Wir stellen verschiedene Seiten der statistischen Lerntheorie vor und erläutern grundlegende Paradigmen und praktische Ansätze, um sie auch für das Reinforcement-Learning einsetzen zu können.

3.1 Das Lernproblem

Die grundlegende Aufgabenstellung beim statistischen Lernen besteht darin, meist natürliche oder technische Phänomene zu beschreiben und Zusammenhänge zu erkennen. Entscheidend ist, dass dabei der induktive Weg gegangen wird, im Gegensatz zu deduktiven Verfahren, die bisher noch weitestgehend dem Menschen vorbehalten sind. Machine-Learning ist daher abzugrenzen von Logik-Systemen, logischen Programmiersprachen, symbolischen Mathematik-Systemen, Expertensystemen oder analytischer, physikalischer Modellierung.

3.1.1 Erkenntnisgewinn durch Induktion

Das Schließen vom Speziellen auf das Allgemeine, von Beobachtungen auf ein Modell der Realität, in der Erkenntnistheorie als Empirismus bezeichnet [42], ist die prinzipielle Methodik, die hier zur Anwendung kommt. Die Schwierigkeiten, die damit einhergehen, sind bereits lange bekannt und gehen zurück auf Aristoteles, William von Occam, John D. Scotus, Thomas von Aquin u.a. und treten in vielen Situationen, besonders in den Naturwissenschaften immer wieder in Erscheinung. Am bekanntesten ist vermutlich die als Konsequenz der 1916 von Einstein entwickelten allgemeinen Relativitätstheorie aufgekommene Diskussion um die Gültigkeit der Aussagen aus der Mechanik, die Jahrhunderte zuvor von Newton postuliert wurden. Ein einfaches wurde durch ein wesentlich komplexeres System ersetzt. Die Komplexität des neuen Modells und der Aufwand seiner Entwicklung und experimentellen Prüfung stehen jedoch in einem so ungünstigen Verhältnis zur praktischen Bedeutung, dass es absolut vernünftig war, lange Zeit vom einfacheren Modell auszugehen. Erst durch neue technische Herausforderungen war es nötig, bessere physikalische Theorien zu entwickeln.

Dieses Problem, Kosten und Präzision einer Theorie gegeneinander abzustimmen, kann auf das Bias-Variance-Dilemma [116, 117, 59, 40, 31, 115] aus der statistischen Lerntheorie zurückgeführt werden. Der Bias entsteht durch Grundannahmen über die Welt, etwa theologisch-philosophische Dogmen wie die Annahme einer globalen Zeit. Dagegen führt das Fehlen von Grundannahmen zu einem großen Erklärungsspielraum, der, wie in diesem Fall offensichtlich ist, einen immensen Experimentierapparat und eine fundierte physikalische Theorie erfordert. Der Bias kann so stark sein, dass erst eine Vielzahl von Beobachtungen zu einer qualitativen Änderung des Ergebnisses führt.

Von großer Bedeutung ist dabei noch eine weitere Beobachtung. Die Qualität einer These über ein Phänomen kann nur daran gemessen werden, wie gut sie in der Lage ist, die Wirklichkeit zu beschreiben. Der Bias jedoch wirkt auf die These selbst, ebenso die Zufälligkeit der zu Grunde gelegten Beobachtungen. Beide Effekte haben dazu geführt, dass in verschiedenen sozialen Umgebungen völlig unterschiedliche Theorien über das Funktionieren des Universums entstanden sind [42], die sich jedoch in ihrer zufriedenstellenden Vorhersage-Qualität über den praktischen Erfahrungshorizont kaum unterscheiden. Letztlich besteht das Ziel des Erkenntnisgewinnes durch Induktion darin, beides zu erreichen, eine Theorie zu finden, die gute Vorhersagen macht und die zu Grunde liegenden Phänomene gut beschreibt.

Zur mathematischen Formalisierung geht Vapnik von einem allgemeinen Risiko-Funktional $R(\alpha)$ aus [117], das von einer Verlustfunktion $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, die von Soll- und Istwert abhängt, sowie vorliegenden Eingabedaten $\mathbf{x} \in X$ (entspricht dem Untersuchungsraum) und Ausgabedaten $y \in Y$ (entspricht den empirischen Beobachtungen) abhängt. Ein Lehrer stellt den Zusammenhang zwischen Eingabe- und Ausgabedaten an den Beobachtungen her. Dabei wird die Wahrscheinlichkeit für das Auftreten bestimmter empirischer Beobachtungen berücksichtigt und neben der korrekten Adaption auch der Bedeutung einer speziellen Beobachtung Rechnung getragen. Auf unsere erkenntnistheoretische Betrachtung bezogen, ist also das Risiko in Newtons Mechanik nicht deutlich größer als das in Einsteins, weil relativistische Effekte in unserer Erfahrung kaum in Erscheinung treten. Dennoch unterscheidet sich die relativistische Mechanik erheblich von der klassischen.

Eine wichtige Problemklasse ist die Regression, die eine Verallgemeinerung der Klassifikation darstellt. Als Verlustfunktion wird zum Beispiel

$$L(y, f(\mathbf{x})) = \|y - f(\mathbf{x})\|^p \tag{3.1}$$

mit einer geeigneten Norm, typischerweise

$$L(y, y') = \|y - y'\|^2 \tag{3.2}$$

gewählt. Das Optimum ist dann der Erwartungswert $f(\mathbf{x}) = \mathbf{E}y = \int yP(y|\mathbf{x})dy$, da

$$\begin{aligned}
\mathbf{E}(f(\mathbf{x}) - y)^2 &= \mathbf{E}\big(f(\mathbf{x})^2 - 2f(\mathbf{x})y + y^2\big) && (3.3)\\
&= f(\mathbf{x})^2 - 2f(\mathbf{x})\mathbf{E}y + \mathbf{E}y^2 = \min && (3.4)\\
\Leftrightarrow f(\mathbf{x})^2 - 2f(\mathbf{x})\mathbf{E}y &= \min && (3.5)\\
\Leftrightarrow 2f(\mathbf{x}) - 2\mathbf{E}y &= 0 && (3.6)
\end{aligned}$$

genau dann wenn $f(\mathbf{x}) = \mathbf{E}y$. Im Falle von

$$L(y, y') = |y - y'| \tag{3.7}$$

ist der Median die Lösung. Die Klassifikation ist ein Spezialfall der Regression, bei dem die Funktionswerte nur die Werte 0 und 1 annehmen können.

3.1.2 Induktive Inferenz

Für Klassifikationsprobleme bezeichnen wir $c \subseteq X$ als das gültige Konzept, falls $\forall x \in c : y(\mathbf{x}) = 1 \wedge \forall \mathbf{x} \notin c : y(\mathbf{x}) = 0$ gilt. Symmetrisch zum Konzept werden Hypothesen $h \subset X$ eingeführt. Während das Konzept c die Realität widerspiegelt, stellt die Hypothese h die Beschreibung selbiger durch eine Lernmaschine dar, die das Klassifikationsproblem lösen soll. Man definiert den Abstand zwischen Hypothese und Konzept durch

$$d(c, h) = \int_{\mathbf{x} \in c \Delta h} P(\mathbf{x})d\mathbf{x}, \tag{3.8}$$

wobei $A\Delta B = A \cup B - A \cap B$. Die Minimierung des Risiko-Funktionals lässt sich also auch als Minimierung des Abstandes zwischen Hypothese und Konzept auffassen. Eine Hypothese

heißt konsistent zu einem Sample $\mathbf{s}_i = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i))$, falls sie alle Daten aus dem Sample korrekt abbildet.

Eine Menge von Konzepten wird als Konzeptklasse C bezeichnet, symmetrisch dazu eine Menge von Hypothesen als Hypothesenraum H. Die Konzeptklasse ist eine Eigenschaft unserer Vorstellung vom vorliegenden Klassifikationsproblem, während der Hypothesenraum eine Eigenschaft der Lernmaschine ist. Eine günstige Wahl des Hypothesenraumes stellt also eine der wichtigsten Designentscheidungen für Lernmaschinen dar.

Diese Definitionen lassen sich für den Regressionsfall entsprechend verallgemeinern.

3.2 Generalisierung, Kapazität und das Bias-Variance-Dilemma

Das bereits angesprochene Bias-Variance-Dilemma lässt sich mit Techniken der statistischen Lerntheorie lösen. Dabei bedeutet Lösen nicht das Eliminieren des Dilemmas, sondern die Wahl einer optimalen Abwägung zwischen Bias und Variance, also Vorurteil und Varianz. Beides soll möglichst klein werden. Wir werden im Folgenden einige Konzepte zur Bewältigung dieses Problems vorstellen und schließlich mit dem Bayes'schen Ansatz eine Alternative aufzeigen, die das Problem tatsächlich aufbricht. Natürlich wird das Dilemma nicht beseitigt, jedoch so umformuliert, dass daraus ein konstruktives Vorgehen wird.

3.2.1 Probably-Approximately-Correct-Learning

Eine Lernmaschine M mit gegebenem Hypothesenraum H ist ein Probably-Approximately-Correct-Learner (PAC), falls für alle $\varepsilon > 0$ und $\delta > 0$ ein $l = l(\varepsilon, \delta, H)$ existiert, so dass

$$P\{d(c, h_l) > \varepsilon\} \quad < \quad \delta. \tag{3.9}$$

Dabei ist l die Anzahl der Trainingsdaten, c das gültige Konzept und $h_l \in H$ die nach l Schritten gefundene Hypothese. Mit anderen Worten, die Wahrscheinlichkeit, dass eine Hypothese zu stark von der Realität abweicht, ist gering.

Ein sehr allgemeines Resultat, da es unabhängig sowohl von der Konzeptklasse als auch dem konkreten Adaptionsalgorithmus, also der Lernmaschine ist, liefert

Satz 1. *[123] Sei M eine Lernmaschine mit Hypothesenraum H, die für l gegebene Beispiele eine konsistente Hypothese aufstellt. Sei ferner $P(\mathbf{x})$ für alle $\mathbf{x}$ gleich groß. Dann ist M ein PAC-Lerner, falls*

$$l \quad \geq \quad \frac{1}{\varepsilon} \log \frac{|H|}{\delta}. \tag{3.10}$$

Beweis: Vorausgesetzt, die nach l Trainingsbeispielen gefundene Hypothese h ist schlecht, es gilt also $d(c, h) > \varepsilon$. Dann ist die Wahrscheinlichkeit, dass M diese trotzdem wählt

$$p_0 \quad \leq \quad (1 - \varepsilon)^l \leq e^{-\varepsilon l}. \tag{3.11}$$

Da nun höchstens alle Hypothesen aus H schlecht sein können, ist die Wahrscheinlichkeit, dass wenigstens eine von Ihnen konsistent mit den ersten l Beispielen ist, höchstens

$$p \quad \leq \quad |H| e^{-\varepsilon l}. \tag{3.12}$$

Da dieser Ausdruck kleiner sein soll als δ, folgt die Behauptung.

□

3.2.2 Kapazitätsmaße

Ist der Hypothesenraum H unendlich groß, kann Satz 1 nicht angewendet werden. Vapnik und Chervonenkis schlagen daher vor, die Größe $\log|H|$ im kontinuierlichen Fall durch die VC-Dimension des Hypothesenraums zu ersetzen. Dabei wird der Hypothesenraum als Funktionsraum

$$H = \{f(\cdot, \alpha)\} \tag{3.13}$$

aufgefasst.

3.2.2.1 Die Growth-Funktion

Seien l Trainingsbeispiele und eine Menge von Funktionen gegeben. Ferner sei $N(\mathbf{x}_1, \ldots, \mathbf{x}_l)$ die Anzahl der Möglichkeiten, die Beispiele $\mathbf{x}_1, \ldots, \mathbf{x}_l$ durch $f(\mathbf{x}, \alpha)$ zu indizieren, also auf verschiedene Art und Weise zu klassifizieren. Dann definiert man die Growth-Funktion

$$G(l) = \log \sup_{\mathbf{x}_1,\ldots,\mathbf{x}_l} N(\mathbf{x}_1, \ldots, \mathbf{x}_l). \tag{3.14}$$

3.2.2.2 Vapnik-Chervonenkis-Dimension

Als eines der herausragendsten Resultate der Lerntheorie gilt nun die Beobachtung [116], die uns direkt zum Begriff der VC-Dimension führt, dass für jede Growth-Funktion entweder

$$G(l) = l \log 2 \tag{3.15}$$

oder

$$G(l) \leq h\left(\log\left(\frac{l}{h} + 1\right)\right) \tag{3.16}$$

gilt. Hat die Growth-Funktion die Gestalt von Gl. 3.16, dann bezeichnen wir h als die VC-Dimension von H, sonst ist die VC-Dimension unendlich.

Eine Menge von Funktionen ist lernbar, falls sie eine endliche VC-Dimension hat. Die VC-Dimension $h = l$ ist die maximale Anzahl an Eingabedaten l, die mit Funktionen aus dem Hypothesenraum in jeder der 2^l möglichen Kombinationen indiziert werden kann. Sind für gegebenes l alle Kombinationen möglich, dann spricht man von der Fähigkeit des Hypothesenraumes, l Punkte zu zerschlagen (shatter). Ist für $l = h$ noch $h\left(\log\left(\frac{l}{h} + 1\right)\right) = l \log 2$, so wächst die rechte Seite offensichtlich asymptotisch linear und damit deutlich schneller als die linke, logarithmisch wachsende Seite. Also ist die Anzahl der möglichen Indikationen für $l > h$ stets kleiner als 2^l.

Die VC-Dimension ist ein sogenanntes Kapazitätsmaß eines Hypothesenraumes, beschreibt also seine Kapazität i.S. von Ausdrucksmöglichkeit.

3.2.2.3 Weitere Kapazitätsmaße

Ein weiteres Kapazitätsmaß für Funktionsmengen stellt die minimale Beschreibungslänge (Minimal-Description-Length, MDL) dar. Die Kapazität wird gerade durch die minimal erforderliche Beschreibungslänge einer entsprechenden Funktion charakterisiert. Bei vielen Klassifikationsverfahren sind VC-Dimension und MDL proportional, es gibt aber auch Ausnahmen. Die Funktionsmenge

$$F = \left\{ f \,\middle|\, f(\mathbf{x}) = \operatorname{sign}\left(\sin\left(\mathbf{w}^T\mathbf{x} - b\right)\right), \mathbf{x}, \mathbf{w} \in \mathbb{R}^2, b \in \mathbb{R} \right\} \tag{3.17}$$

hat zum Beispiel VC-Dimension $h = \infty$, verfügt aber nur über drei Parameter.

Für Regressionsprobleme wird die VC-Dimension so verallgemeinert, dass man einen weiteren Parameter einführt, der den kontinuierlichen Charakter des Wertebereichs berücksichtigt. Weitere Verallgemeinerungsmöglichkeiten sind etwa durch die Covering-Numbers, die Fat-Shattering-Dimension und die Pseudo-Dimension gegeben [2, 18].

3.2.2.4 Nicht-Falsifizierbarkeit

Das Prinzip der Nicht-Falsifizierbarkeit führt wieder zurück auf die erkenntnistheoretische Ebene und eröffnet eine Möglichkeit, die Bedeutung des Konzeptes der VC-Dimension zu überschauen. Popper bezeichnet eine Wissenschaft als gerechtfertigt (justifiable), wenn es klare Regeln gibt, wie Hypothesen falsifiziert werden können [75]. Das ist dann der Fall, wenn es Ergebnisse von Experimenten geben könnte, die innerhalb des Paradigmas dieser Wissenschaft nicht erklärt werden können.

Bezogen auf das Konzept der VC-Dimension handelt es sich dabei gerade um die Gegenüberstellung von endlicher zu unendlicher VC-Dimension. Hat der Hypothesenraum, der dem Paradigma entspricht, eine unendliche VC-Dimension, dann kann für jede gegebene Menge an Beobachtungen eine Hypothese aufgestellt werden, die ein Element des Hypothesenraumes ist. Da der Hypothesenraum demnach innerhalb des Beobachtungsbereiches nicht falsifiziert werden kann, ist er ungeeignet, da nur mit einer endlichen VC-Dimension ab einer bestimmten Anzahl an Beobachtungen gewährleistet wird, dass nur eine begrenzte Kombination von Samples möglich ist.

3.2.3 Strukturelle Risiko-Minimierung

Die strukturelle Risiko-Minimierung ist eine Meta-Lernmethode, die die Kapazität der Hypothesenräume berücksichtigt und hinsichtlich des zu lösenden Problems optimiert, so dass die bzgl. der beobachteten Daten optimierte Hypothese auch einen möglichst geringen Generalisierungsfehler aufweist. Je größer die Kapazität, desto mehr Beispiele können ohne Fehler adaptiert werden, aber desto geringer ist auch die Generalisierungsfähigkeit des entstehenden Klassifikators. Eine niedrige Kapazität führt zu einer geringeren Beschreibungsfähigkeit

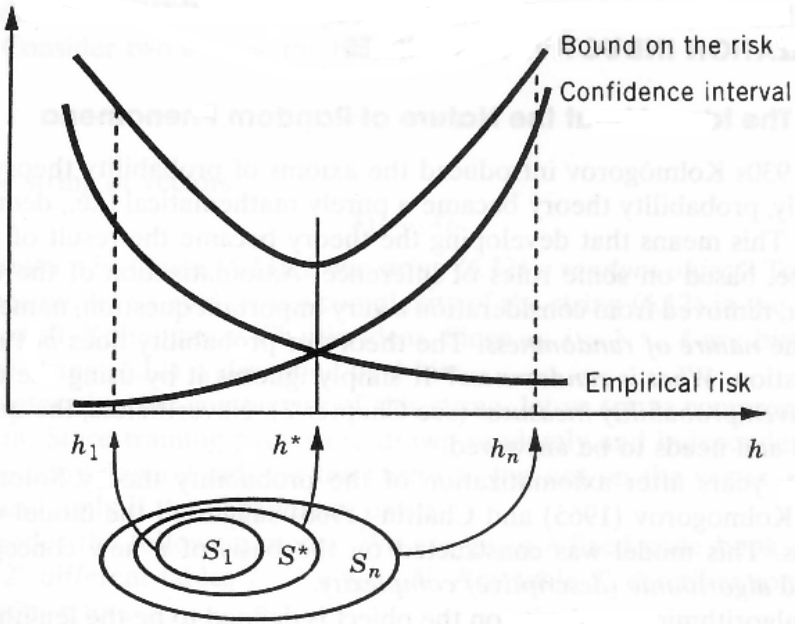


Abbildung 3.1: Strukturelle Risiko-Minimierung: Der Hypothesenraum muss so groß wie nötig gewählt werden, um möglichst viele Konzepte aufzunehmen und einen kleinen Bias zu ermöglichen, aber so klein wie möglich, damit auch tatsächlich eine möglichst geeignete Hypothese gefunden wird und die Varianz ebenfalls klein ist. Die steigende Kapazität führt zu einem kleineren Fehler auf den Beobachtungen, aber auch zu einem größerem Kapazitätsterm, der zur Ermittlung des Generalisierungsfehlers berücksichtigt werden muss. Quelle: [88].

der Hypothesen aus dem Hypothesenraum. Also werden entsprechend weniger Trainingsbeispiele gut adaptiert. Dafür ist die Wahrscheinlichkeit, dass andere Eingabedaten korrekt

klassifiziert werden, größer (siehe Abb. 3.1).

Im Einzelnen wurden verschiedene Schranken für das Risiko-Funktional in Abhängigkeit von VC-Dimension und Anzahl der Trainingsbeispiele gefunden, die etwa in [116, 17] aufgeführt sind. Für linear klassifizierbare Konzepte ist folgendes relevante Resultat interessant. Das Theorem ist von zentraler Bedeutung für den Nachweis der Generalisierungsfähigkeit von Support-Vector-Machines, die in Abschn. 3.4.2 vorgestellt und in Abschn. 6.5 für das Reinforcement-Learning eingesetzt werden.

Satz 2. *(Vapnik) [116] Sei die Konzeptklasse C auf einer Domäne $X \subset \mathbb{R}^n$, deren Elemente sich innerhalb einer Hyperkugel mit Radius R befinden und in der linear separierbare Konzepte ausschließlich mit Margin δ^* konstruiert werden können, gegeben. Dann ist die VC-Dimension*

$$\mathbf{VCD}(C) \quad = \quad \min\left(n, \left[\frac{R^2}{(\delta^*)^2}\right]\right) + 1. \tag{3.18}$$

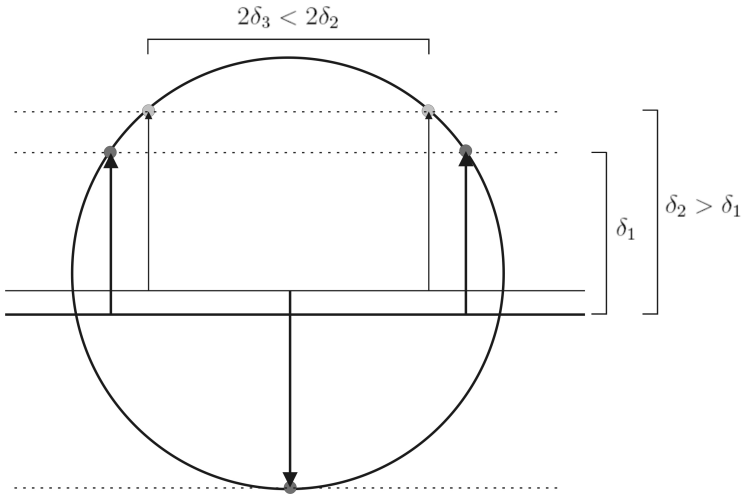


Abbildung 3.2: Beschränkung der VC-Dimension mit Hilfe des Margins: Je größer der Margin ($\delta_2 > \delta_1$), desto kleiner ist die VC-Dimension der Menge der Konzepte, die mit einem linearen Klassifikator diesen Margin erfüllen. Im besten Fall repräsentieren die Vektoren Punkte eines gleichmäßigen Simplexes. Vergrößert man dann den Maximum-Margin, müssen mindestens zwei Punkte enger zusammenrücken ($\delta_3 < \delta_2$) und damit verringert sich die Anzahl der möglichen Indikationen, also auch die VC-Dimension.

Das Prinzip der SRM besteht schließlich darin, eine Struktur

$$S_0 \quad \subset \quad S_1 \subset \ldots \subset S_i \subset \ldots \tag{3.19}$$

von Hypothesenräumen zu definieren und denjenigen Hypothesenraum auszusuchen, der bzgl. des erwarteten Generalisierungsfehlers am besten ist. Diese Einschätzung kann auf Grund theoretischer Fehlerschranken oder auf der Grundlage experimenteller Ergebnisse auf Validierungsdaten erfolgen.

3.3 Der Bayes'sche Ansatz

Ein zum Konzept der strukturellen Risiko-Minimierung entgegengesetzter Ansatz zur Generalisierung stellt das Bayes'sche Vorgehen dar. Im Gegensatz zur Regularisierung, bei der durch die Anwendung von Regularisierungstermen oder die Einschränkung der Hypothesenräume die tatsächliche Zielfunktion verändert wird, erfolgt beim Bayes'schen Lernen ein konstruktives Vorgehen. Das Bias-Variance-Dilemma wird dadurch aufgebrochen, dass die beiden Elemente des Dilemmas, Bias und Varianz, positiv interpretiert werden.

3.3.1 A priori und a posteriori Verteilung

Grundlage des Bayes'schen Lernens ist die Bayes-Regel [10, 58, 40, 59]

$$P(A|B)P(B) = P(B|A)P(A), \tag{3.20}$$

nach der sich die bedingte Wahrscheinlichkeit einer Zufallsvariable A gegeben B unmittelbar aus der symmetrischen bedingten Wahrscheinlichkeit für B gegeben A bestimmen lässt. Wenn wir also A als Zufallsvariable über der Menge aller möglichen Hypothesen und B über der Menge möglicher Beobachtungen definieren, folgt

$$P(h|Z) = \frac{P(Z|h)P(h)}{P(Z)}, \tag{3.21}$$

mit h der Hypothese und Z den Beobachtungen. Der Ausdruck $P(Z|h)$, der als Likelihood bezeichnet wird, ist im Prinzip leicht zu berechnen, führt also höchstens zu einem Problem des erforderlichen Rechenaufwandes, $P(h)$ und $P(Z)$ sind die a priori Wahrscheinlichkeiten für Hypothese und Modell. Um zwischen der Eignung zweier Hypothesen h_1 und h_2 zu entscheiden, genügt es offensichtlich

$$P(h_1|Z) > P(h_2|Z) \tag{3.22}$$

$$\Leftrightarrow P(Z|h_1)P(h_1) > P(Z|h_2)P(h_2) \tag{3.23}$$

zu betrachten. Die Wahl von $P(h)$ entspricht der Bestimmung des sogenannten Priors für jede mögliche Hypothese, was einer konstruktiven Beschreibung des Bias entspricht.

Geht man von der vereinfachenden Annahme aus, dass alle Hypothesen gleich wahrscheinlich sind, kollabiert die Methode zum Maximum-Likelihood-Ansatz [40, 59], da die Hypothese gewählt wird, für die mit höchster Wahrscheinlichkeit die beobachteten Daten hätten generiert werden können. Da dieses Vorgehen jedoch den Prior unberücksichtigt lässt, ist es lediglich eine Näherung an die Bayes-Lösung. Dennoch ist die Methode konsistent, asymptotisch erwartungstreu und asymptotisch effizient [114].

3.3.2 Bayes'sche Inferenz

Der oben beschriebene Bayes-Ansatz bestimmt den Modalwert der a posteriori Verteilung der Hypothesen bei gegebenen Beobachtungen, also die Hypothese, deren Wahrscheinlichkeit am größten ist. Ist der Hypothesenraum unendlich groß und lässt sich mit Hilfe einer endlichen Anzahl kontinuierlicher Parameter θ beschreiben, kann die Methode verwendet werden, um die wahrscheinlichste Parameterwahl zu bestimmen, indem

$$P(\theta|Z,H) = \frac{P(Z|\theta,H)P(\theta|H)}{P(Z|H)} = \max \tag{3.24}$$

$$\Leftrightarrow P(Z|\theta,H)P(\theta|H) = \max \tag{3.25}$$

bzgl. θ maximiert wird. Dieses Vorgehen ist jedoch innerhalb der Bayes-Philosophie ebenfalls nur eine Näherung, da der Modalwert im Allgemeinen nicht der geeignetste Schätzer der zu Grunde liegenden Zufallsvariable ist und wird als Maximum-A-Posteriori-Methode (MAP) bezeichnet.

Als vollständige korrekte Inferenz einer neuen Beobachtung $\mathbf{z}$ ergibt sich in der Tat die prediktive Verteilung

$$P(\mathbf{z}|Z,H) = \int_\theta P(\mathbf{z}|\theta,H)P(\theta|Z,H)d\theta, \tag{3.26}$$

die eine vollständige Dichtefunktion liefert, mit der die a posteriori Schätzung vorgenommen werden kann.

Dies ist in der Praxis zumeist jedoch nur aufwändig zu berechnen, daher wird als alternative und erwartungstreue Schätzung der Parameter der Erwartungswert

$$\begin{aligned} \mathbf{E}(\theta|Z,H) &= \int_\theta P(\theta|Z,H)\theta d\theta && (3.27)\\ &= \int_\theta \frac{P(Z|\theta,H)P(\theta|H)}{P(Z|H)}\theta d\theta && (3.28)\\ &= \frac{1}{P(Z|H)}\int_\theta P(Z|\theta,H)P(\theta|H)\theta d\theta && (3.29) \end{aligned}$$

über alle möglichen Hypothesen gewählt, der näherungsweise der Schätzung gemäß Gl. 3.26 entspricht und sich im Bayes'schen Kontext als Gewinn gegenüber Gl. 3.24 herausstellt.

Die Bestimmung der erwarteten Hypothese kann in der Praxis nach wie vor ein sehr aufwändiger Vorgang sein, da, abhängig von der Problemstellung, ein nicht triviales Integral berechnet werden muss. Aus diesem Grund sind sogenannte konjugierte Prior bzw. Paare von Likelihood und konjugiertem Prior identifiziert worden, der Begriff des konjugierten Priors wurde in [79] eingeführt. Zu einer Likelihood-Wahrscheinlichkeitsverteilung $P(Z|\theta,H)$ ist eine a priori Verteilung $P(\theta|H)$ ein konjugierter Prior, falls die a posteriori Verteilung $P(\theta|Z,H)$ zur gleichen Klasse gehört wie $P(\theta|H)$.

Der Vorteil ist, dass für eine Vielzahl von a priori Verteilungen sich die a posteriori Verteilungen durch relativ einfache arithmetische Operationen auf ihren Parametern bestimmen lassen. In Folge dessen sind auch Modalwert und Erwartungswert sowie weitere statistische Kenngrößen wie die Varianz leicht zu berechnen. Der Nachteil besteht offensichtlich in der eingeschränkten Parametrisierung sowohl des Likelihoods als auch des Priors. Ist der Likelihood bekannt, so muss die Wahl eines geeigneten konjugierten Priors und deren Parameter nicht zwangsläufig mit der tatsächlich angenommenen a priori Wahrscheinlichkeitsverteilung übereinstimmen und beschreibt in diesem Fall nur eine Näherung.

3.3.3 Unsicherheit

Einer der entscheidenden Vorteile der Bayes'schen Inferenz besteht in der Verfügbarkeit der Unsicherheit sowohl der Schätzer θ als auch der neuen Beobachtungen $\mathbf{z}$, die sich durch die Varianz der Verteilungen $P(\theta|Z,H)$ und $P(\mathbf{z}|Z,H)$ ausdrücken lässt. Statistische Unsicherheit ist ein Thema von zentraler Bedeutung in vielen Anwendungsgebieten einschließlich des Machine-Learning. Es ist als Selbstverständlichkeit akzeptiert, dass jede Messung in der Natur und jede Schlussfolgerung, die aus Messergebnissen stammt, mit einer Unsicherheit behaftet sind. Die International Organization for Standardization (ISO) definiert Unsicherheit als

> a parameter, associated with the result of a measurement, that characterizes the dispersion of the values that could reasonably be attributed to the measurand [118].

Die Unsicherheit eines Messwertes ist demnach also ein mit dem Ergebnis einer Messung assoziierter Parameter, der die Streuung der Werte charakterisiert, die in sinnvoller Weise für die Messgröße angenommen werden können. Statistische Unsicherheit entsteht dadurch, dass die gemessene Größe einem stochastischen Rauschen unterliegt oder sogar eine inhärente Stochastizität aufweist, weil sie etwa noch von anderen Aspekten abhängt. Quantenmechanisch ist bereits jede Messung lediglich stochastisch, sie kann aber auch stochastisch sein, weil ihr ein nicht exakt wiederholbares Experiment zu Grunde liegt, bei dem sich Nebenbedingungen verändern. In diesem Fall muss eine ausreichende Zahl von Beobachtungen gemacht werden, damit die geschätzten Messwerte hinreichend genau sind.

Bei einer frequentistischen Analyse von Messungen $x_1,\ldots,x_l$ einer Größe x betrachtet man dazu den Mittelwert $\bar{x} = \frac{1}{l}\sum_{i=1}^{l} x_i$ und die Varianz $\sigma_x^2 = \frac{1}{l-1}\sum_{i=1}^{l}(x_i-\bar{x})^2$ dieser Messungen. Durch den bereits verwendeten Freiheitsgrad $\bar{x}$ als Schätzer für $\mathbf{E}x$ ergibt

sich die Notwendigkeit des modifizierten Vorfaktors zur Bestimmung eines erwartungstreuen Schätzers für die Varianz. Für den Erwartungswert ist dieser wie oben gegeben. Nun wird die quadratische Unsicherheit über $\mathbf{E}x$, die proportional zu $\mathbf{var}x$ ist, mit steigendem l sinken, sie ergibt sich zu $\sigma^2_{\text{unc}} = \frac{1}{l}\sigma^2_x$ und entspricht der Varianz des geschätzten Mittelwertes $\bar{x}$ bei mehreren Samples der Größe l unter der Annahme, dass für jedes Sample die Varianz σ^2_x geschätzt wird. Der geschätzte Absolutfehler, also das, was wir tatsächlich als (absolute) Unsicherheit bezeichnen, beträgt dann σ_{unc} und ist proportional sowohl zu σ_x, der Standardabweichung der Messungen, als auch zu $1/\sqrt{l}$ und sinkt damit mit der Wurzel der Anzahl der Stichproben. Diese Erkenntnis gilt qualitativ auch für die Bayes'sche Perspektive und stellt die Grundlage für die in Kap. 9 vorgenommenen Betrachtungen dar.

Die Unsicherheit liefert schließlich ein Maß für die Vertrauenswürdigkeit des Ergebnisses. Ist die Varianz sehr groß, kann der tatsächliche Wert für θ bzw. $\mathbf{z}$ weit entfernt vom Erwartungswert liegen, ist sie niedrig, so ist die Qualität des Schätzers besser. Die Qualität ist daher erst durch die Angabe der Unsicherheit zu bewerten. Darüber hinaus kann auch die konkrete Gestalt der Verteilungen wichtige Auskünfte geben. So können Erwartungswert und Modalwert weit auseinanderliegen oder der Erwartungswert nur eine geringe Wahrscheinlichkeit aufweisen. Die Bayes'sche Inferenz liefert also ein mächtiges Instrumentarium zur sorgfältigen Analyse der Ergebnisse.

3.3.4 Unsicherheitspropagation

Die Unsicherheit der Messwerte kann mittels Unsicherheitspropagation bis zu den Schlussfolgerungen weitergeleitet werden. Das Prinzip der Unsicherheitspropagation [19], genauer der Gauß'schen Propagation von Unsicherheiten oder der Gauß'schen Fehlerfortpflanzung, basiert auf einer Taylor-Entwicklung

$$f(x+\sigma x) \quad = \quad f(x) + \frac{df(x)}{dx}\sigma x + \frac{d^2 f(x)}{2dx^2}(\sigma x)^2 + \dots \tag{3.30}$$

ersten Grades um den geschätzten Punkt. Dabei wird die quadratische Unsicherheit der Werte $f(\mathbf{x})$ im multivariaten Fall, also für $f : \mathbb{R}^m \to \mathbb{R}^n$, gegeben die Unsicherheit ihrer Argumente $\mathbf{x}$ gemäß

$$\mathbf{Cov}(f,f) \quad = \quad D\mathbf{Cov}(\mathbf{x},\mathbf{x})D^T \tag{3.31}$$

bestimmt, wobei $D = \frac{\partial f}{\partial \mathbf{x}}$ die Jacobi-Matrix von f nach ihren Argumenten $\mathbf{x}$ beschreibt. Dabei ist $\mathbf{Cov}(\mathbf{x},\mathbf{x})$ die Kovarianzmatrix der Argumente, die wiederum die Unsicherheit von $\mathbf{x}$ enthält. Der Funktionswert f verfügt dann über die symmetrische und positiv definite Kovarianzmatrix und Unsicherheit $\mathbf{Cov}(f,f)$. Dieses Konzept ist strikt nur dann anwendbar, wenn die Messungen normalverteilt sind und f linear von $\mathbf{x}$ abhängt. Die Anforderung an den Typ der Verteilung lässt sich jedoch abschwächen zur Forderung von Symmetrie. Die Normalverteilung ist nur dann erforderlich, wenn bei der Interpretation der Unsicherheiten Perzentile verwendet werden, die der Normalverteilung zu Grunde liegen.

3.3.5 Frequentistische und Bayes'sche Statistik

Die statistische Lerntheorie beruht letztlich auf den Gesetzmäßigkeiten der Statistik. Daher ist es sinnvoll, auf den paradigmatischen Unterschied klassischer (frequentistischer) und Bayes'scher Statistik hinzuweisen. Die klassische Statistik kennt das Konzept der a priori und a posteriori Verteilungen nicht und ersetzt diese durch Null- und Arbeitshypothese.

Anstatt von einem zuvor festgelegten Prior auszugehen, der beschreibt, was der Anwender glaubt, wie die Wirklichkeit aussieht, wenn er noch keine Beobachtungen gemacht hat, um dann auf die a posteriori Verteilung zu schließen, wird in der klassischen Statistik ohne Prior argumentiert und die Verteilung ausschließlich auf der Grundlage der Beobachtungen

bestimmt. Der Anwender legt dabei eine Nullhypothese θ_0 fest, welche er für die wahrscheinlichste hält oder deren Gültigkeit er überprüfen möchte. Ist nun

$$\int_{Z' \in D_Z} P(Z'|\theta_0, H) dZ' \quad > \quad \alpha \tag{3.32}$$

für ein zuvor festgelegtes α, dann wird die Nullhypothese angenommen. Dabei ist D_Z so zu wählen, dass die tatsächlich beobachteten Daten Z eine untere Grenze in D_Z kennzeichnen in dem Sinne, dass Z am Besten mit der Nullhypothese vereinbar ist. Es wird also überprüft, wie wahrscheinlich es ist, dass Z oder eine noch extremere Beobachtung gemacht wird.

Man erkennt sofort die beiden entscheidenden Nachteile dieses frequentistischen Herangehens. Zum Einen hat die auf diese Weise gewonnene Erkenntnis lediglich qualitative Aussagekraft, denn anstelle einer vollständigen a posteriori Verteilung erhält man eine binäre Entscheidung für oder gegen die Nullhypothese, diese hängt zudem von der Wahl von α ab. Zum Anderen bezieht man sich auf eine Fragestellung, die invers zu der tatsächlich zu beantwortenden ist. Schließlich ist man an der Bestimmung von $P(\theta|Z, H)$ und nicht an der von $P(Z|\theta, H)$ interessiert.

3.3.6 Kapazität und die Bayes'sche Perspektive

Obwohl das SRM-Prinzip mit der Verwendung von Kapazitätsmaßen und die Bayes'sche Inferenz unterschiedlichen Paradigmen folgen, führen sie letztlich fast immer zum gleichen Ziel. Im Abschnitt 3.4.2.6 wird z.B. die Bayes'sche Perspektive der Support-Vector-Machines skizziert.

Wir wollen die Äquivalenz an einem einfachen Beispiel zeigen. Dieses besteht in der Schätzung des Erwartungswertes einer skalaren Zufallsvariable X an Hand von l Beobachtungen. Die Minimierung des quadratischen Fehlers

$$L(x) \quad = \quad \frac{1}{l} \sum_{i=1}^{l} (x - x_i)^2 = \min \tag{3.33}$$

führt zur Schätzung

$$x \quad = \quad \frac{1}{l} \sum_{i=1}^{l} x_i, \tag{3.34}$$

also durch den Mittelwert der Daten. Versieht man die Fehlerfunktion zusätzlich mit einem Regularisierungsterm $\Omega(x)$ zur Forcierung kleiner Werte für x, indem

$$\hat{L}(x) \quad = \quad L(x) + \lambda \Omega(x) \tag{3.35}$$

$$\quad = \quad L(x) + \lambda x^2 = \min \tag{3.36}$$

gewählt wird, ergibt sich die Lösung

$$x \quad = \quad \frac{1}{(1+\lambda) l} \sum_{i=1}^{l} x_i. \tag{3.37}$$

In der Bayes'schen Perspektive hingegen könnte man annehmen, dass X Gauß-verteilt mit bekannter Varianz σ ist, der Likelihood sich also als

$$P(x_1, \ldots, x_l|x) \quad = \quad \prod_{i=1}^{l} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2\sigma^2}} \tag{3.38}$$

schreiben lässt. Da sich das Maximum des Produktes in Gl. 3.38 durch Logarithmieren zu

$$\log P(x_1,\ldots,x_l|x) = -l\log\left(\sigma\sqrt{2\pi}\right) - \sum_{i=1}^{l}\left(\frac{(x-x_i)^2}{2\sigma^2}\right) = \max \tag{3.39}$$

$$\Leftrightarrow \sum_{i=1}^{l} -(x-x_i)^2 = \max \tag{3.40}$$

ergibt, ist die Maximum-Likelihood-Lösung, die der frequentistischen Perspektive entspricht, identisch mit der Schätzung in Gl. 3.34. Das gleiche Ergebnis erhält man auch bei Wahl eines Gauß-Priors für x mit Erwartungswert $\mu_0 = 0$ und Varianz $\sigma_0 = \infty$. Aus diesem Grund wird dieser Prior als uninformiert bezeichnet. Wird jedoch $\sigma_0 = \sigma/\sqrt{\lambda l}$ gewählt, erhält man die Lösung aus Gl. 3.37. Dies ergibt sich unmittelbar aus

$$P(x|x_1,\ldots,x_l) = \frac{P(x_1,\ldots,x_l|x)P(x)}{P(x_1,\ldots,x_l)} \tag{3.41}$$

$$= \frac{\prod_{i=1}^{l}\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-x_i)^2}{2\sigma^2}}\frac{1}{\sigma_0\sqrt{2\pi}}e^{-\frac{(0-x)^2}{2\sigma_0^2}}}{P(x_1,\ldots,x_l)} = \max \tag{3.42}$$

$$\Leftrightarrow -\sum_{i=1}^{l}(x-x_i)^2 - \frac{\sigma^2x^2}{\sigma_0^2} = \max \tag{3.43}$$

$$\Leftrightarrow x = \frac{1}{l+\sigma^2/\sigma_0^2}\sum_{i=1}^{l}x_i \tag{3.44}$$

und der Symmetrie der Gauß-Verteilung, so dass Modalwert und Erwartungswert identisch sind.

3.4 Grundlegende Verfahren

In diesem Abschnitt werden grundlegende technische Umsetzungen obiger Prinzipien erläutert. Wir stellen neuronale Netze und Kernel-Machines vor, die Basis für die in der vorliegenden Arbeit vorgestellten Ausführungsformen der Rewards-Regression sind (siehe Kap. 6 und 7).

3.4.1 Neuronale Netze

Kein anderes Modell hat das Machine-Learning vergleichbar stark geprägt wie das der neuronalen Netze [41, 43]. Historisch stellen sie die Basis für die Motivation des gesamten Forschungsfeldes dar und frühe Modelle sind das Ergebnis der ersten Schritte, intelligentes Verhalten oder deren Grundlagen der Biologie technisch umzusetzen. Neuronale Netze werden oftmals als eine Zusammenfassung von Funktionsapproximatoren bezeichnet. I.e.S. jedoch haben sie spezielle Eigenschaften und grenzen sich von anderen Verfahren ab.

Im Bereich des Unsupervised-Learning sind einige der wichtigsten Beiträge neuronale Assoziativspeicher wie die Hopfield-Netze [45] und selbstorganisierende Karten wie die Kohonen-Netze [53] und Neural-Gas [64]. Im Folgenden werden wir uns ausschließlich Modellen für neuronale Netze zum Supervised-Learning zuwenden.

3.4.1.1 Aufbau neuronaler Netze

Die Grundbausteine eines jeden neuronalen Netzes sind Neuronen und Synapsen, die die Neuronen miteinander verbinden. Neuronen sind charakterisiert durch eine Aktivierungsfunktion $h : \mathbb{R} \to \mathbb{R}$ sowie der eingehenden und ausgehenden Synapsen, die Synapsen wiederum durch ihr Gewicht. Künstliche Neuronen sind dabei eng an das biologische Vorbild angelehnt. Jede

eingehende Synapse kann einen elektrischen Impuls an das Neuron senden. Das Neuron integriert alle diese Impulse auf, wertet die Aktivierungsfunktion aus und leitet den entstehenden Impuls an alle ausgehenden Synapsen weiter. Im Gegensatz zu Spiking-Neuron-Modellen [32] gehen wir von der Vereinfachung aus, dass Impulse (genauer Impulsfolgen) durch reelle Zahlen ausgedrückt und zu diskreten Zeitpunkten ausgewertet werden.

Das Signal, das Neuron j also zu einem bestimmten Zeitpunkt sendet, ergibt sich aus

$$\mathbf{x}_j = h\left(\sum_{i=1}^{d_j} \mathbf{w}_{j_i,j}\mathbf{x}_{j_i}\right), \tag{3.45}$$

wobei j_i das $i$te Neuron, das mit einer eingehenden Synapse mit dem Neuron j verbunden ist, d_j die Anzahl dieser Neuronen, $\mathbf{x}_{j_i}$ die Aktivierung des Neurons j_i und $\mathbf{w}_{j_i,j}$ das Gewicht der verbindenden Synapse bezeichnen. Einige Neuronen werden als Eingabe-, andere als Ausgabeneuronen ausgezeichnet. Diese dienen zur Kommunikation mit der Außenwelt. Bei Klassifikations- oder Regressionsaufgaben werden also die Eingabedaten an die Eingabeneuronen angelegt und die Ergebnisse an den Ausgabeneuronen abgelesen. Alle anderen Neuronen werden als versteckte Neuronen bezeichnet.

Ein Cluster bezeichnet eine Gruppe von Neuronen mit ähnlichen Eigenschaften. Sie verfügen insbesondere über die gleiche Aktivierungsfunktion. Cluster werden untereinander mit Konnektoren verbunden, die analog eine Gruppe von Synapsen darstellt. Jedes Neuron des einen Clusters ist dabei mit jedem Neuron des anderen mit einer Synapse aus dem verbindenden Konnektor verbunden. Cluster und Konnektoren können über weitere Eigenschaften verfügen, die ihre einzelnen Bestandteile nicht aufweisen können.

Diese Abstraktion ermöglicht die vektorielle Schreibweise von Gl. 3.45 als

$$\mathbf{x}^j = h\left(W_{i,j}\mathbf{x}^i\right). \tag{3.46}$$

Dabei bezeichnen $\mathbf{x}^i$ den Cluster i und $W_{i,j}$ die Matrix mit den Synapsengewichten des Konnektors, der die Cluster i und j miteinander verbindet.

3.4.1.2 Grundlegende Modelle

Das einfachste neuronale Netz ist das sogenannte Perceptron [85]. Es besteht aus einem einzigen Konnektor, der Eingabe- und Ausgabecluster miteinander verbindet. Das Multi-Layer-Perceptron (Feed-Forward-Netz) ist eine Verallgemeinerung des Perceptrons. Es besteht aus Eingabe- und Ausgabecluster sowie aus N versteckten Clustern, die sequentiell miteinander verbunden sind. Daher bezeichnet man die Cluster hier auch als Schichten. Während das Perceptron, abhängig von der benutzten Aktivierungsfunktion, nur einfache Funktionen abbilden kann, sind Feed-Forward-Netze so mächtig, dass für eine breite Klasse von Aktivierungsfunktionen bereits mit einer versteckten Schicht beliebige stückweise Borel-messbare und damit insbesondere stetige Funktionen beliebig genau approximiert werden können [46].

Als Aktivierungsfunktionen sind prinzipiell beliebige einstellige differenzierbare Funktion erlaubt. Meistens werden jedoch sigmoide Funktionen verwendet, wie etwa

$$h(x) = \tanh(ax + b). \tag{3.47}$$

Sie sollten die Eigenschaft haben, dass $\forall y : h(x) \to c_{\max} > h(y)$ für $x \to \infty$ und symmetrisch $\forall y : h(x) \to c_{\min} < h(y)$ für $x \to -\infty$. Im Falle der Aktivierungsfunktion in Gl. 3.47 gilt die universelle Approximierbarkeit [46].

3.4.1.3 Lernverfahren

Für eine gegebene Menge von Beobachtungen $\mathbf{x}_i \in X$ und $y_i \in Y$ gilt es nun, eine Konfiguration eines neuronalen Netzes zu finden, so dass eine Funktion $f : X \to Y$ approximiert wird. Zur Konfiguration gehören dabei sowohl die Gewichte $W_{i,j}$ als auch die Parameter der

Aktivierungsfunktionen (in Gl. 3.47 z.B. die Variablen a und b), die für jedes Neuron verschieden sein können. Die Tatsache, dass die Aktivierungsfunktionen im Allgemeinen nicht-linear sind, versperrt die Anwendbarkeit einer ganzen Reihe effizienter Optimierungsalgorithmen. Man bedient sich daher des Gradientenabstiegsverfahrens, das sehr allgemein einsetzbar ist. Dabei wird für jeden Ausgabecluster i eine charakteristische Zielfunktion $L^i_\omega : X \times Y \to \mathbb{R}$ gewählt und der Ausdruck

$$\sum_{i=1}^{d_a} L^i_\omega \quad = \quad \sum_{i=1}^{d_a} \sum_{j=1}^{l} L^i_\omega(\mathbf{x}_j, y_j) = \min \tag{3.48}$$

minimiert, d_a bezeichnet die Anzahl der Ausgabecluster und l die Anzahl der verfügbaren Beobachtungen. Dies geschieht durch Differenzieren nach den Parametern ω von f, die die Übergangsmatrizen und die Parameter der Aktivierungsfunktionen umfassen, gemäß

$$\Delta\omega \quad = \quad \sum_{i=1}^{d_a} \sum_{j=1}^{l} \frac{\partial}{\partial\omega} L^i_\omega(\mathbf{x}_j, y_j) \tag{3.49}$$

und Durchführen eines Gradientenabstiegs in Richtung $-\alpha\Delta\omega$ mit Lernschrittweite α.

Zur effizienten und modularen Berechnung des Gradienten wird i.d.R. eine Variante des Back-Propagation-Algorithmus [121, 86, 43] benutzt, der speziell für das Lernen neuronaler Netze entwickelt wurde.

Das Hauptproblem der gradientenbasierten Optimierung in neuronalen Netzen besteht in der Nicht-Konvexität der Fehlerfunktion, die aus der Nicht-Linearität der Aktivierungsfunktionen resultiert. Daher kann das Verfahren bald in ein lokales Minimum laufen, an dem der Gradient zwar verschwindet, aber nicht das globale Minimum gefunden wurde. Da das Problem der globalen nicht-konvexen Optimierung NP-vollständig ist [89], besteht geringe Aussicht, es mit Varianten des Gradientenverfahrens prinzipiell zu lösen. Dennoch gibt es eine Reihe von Heuristiken, die gute lokale Minima finden.

Grundsätzlich unterscheidet man zwischen den Batch- und Pattern-by-Pattern-Verfahren. Beim Batch-Learning wird der Gradient der Fehlerfunktion in jedem Lernschritt vollständig berechnet, während beim Pattern-by-Pattern-Learning lediglich der Beitrag einer einzelnen Beobachtung am Gradienten berücksichtigt wird. Dies kann auch auf einer kleinen Gruppe von Beobachtungen geschehen, deren Größe durch die sogenannte Batch-Size bestimmt wird. Erfolgt die Auswahl dieser Gruppe stochastisch, ist es bedeutend wahrscheinlicher, dass der Algorithmus das globale Minimum findet. Dies und die schnellere Konvergenz sind entscheidende Vorteile der stochastischen Optimierungsverfahren.

Weiteres Potenzial zur Verbesserung der Konvergenz besteht in der Kombination mit dem Newton-Verfahren [28], der Anwendung von Verfahren höherer Ordnungen oder des Vario-Eta-Verfahrens [68], im Einsatz von Momentum-Termen oder in der Kombination mit binärer Suche wie beim Resilient-Propagation-Verfahren [84].

Vielfältige Maßnahmen zur Regularisierung in neuronalen Netzen wurden entwickelt, etwa durch Node-Pruning und Weight-Pruning sowie durch Erhöhen der Anzahl der Ausgabeneuronen, Einbringen von zufälligen Ausgabewerten sowie künstlichem Rauschen in den Eingabegrößen und Cleaning-Techniken [68]. Auf drei Varianten des Gradientenabstiegs, die auch als Regularisierungstechnik eingesetzt werden, soll an dieser Stelle näher eingegangen werden.

3.4.1.4 Weight-Decay

Beim Weight-Decay werden die Fehlerfunktionen so modifiziert, dass hohe Gewichte der Synapsen bestraft werden, indem

$$\hat{L}^i_\omega = L^i_\omega + \lambda\Omega(\omega) \tag{3.50}$$

gesetzt wird. Der Parameter λ steuert den Einfluss des Weight-Decays. Diese Technik forciert einfachere und, im Falle der Anwendung von Gl. 3.47, Lösungen mit einer stärker ausgeprägten linearen Komponente. Die Wahl von λ entspricht also der Wahl der geeigneten Struktur im Paradigma der strukturellen Risiko-Minimierung.

3.4.1.5 Regularisierung durch stochastisches Back-Propagation und Vario-Eta

Stochastische Optimierungsverfahren wie das Pattern-by-Pattern-Learning oder mit einer kleinen Batch-Size führen, unter der Voraussetzung, dass die Lernschrittweite im Limit nicht weiter reduziert wird, zu einer veränderten Zielfunktion. Das Vario-Eta-Verfahren modifiziert diese so, dass die zweiten Ableitungen der Fehlerfunktion nach den Parametern des neuronalen Netzes zur Fehlerfunktion hinzuaddiert werden. Dies hat den Effekt, dass wenig ausgeprägte, also spitze Minima stärker bestraft werden als flache. Flache Minima entsprechen mit höherer Wahrscheinlichkeit einer angemessenen Lösung [68]. Dabei wird die Lernschrittweite für jedes einzelne Gewicht so modifiziert, dass sie mit dem Reziproke der Varianz der Gradienten des Batch-Datensatzes gewichtet wird. Dies führt außerdem dazu, dass das sogenannte Vanishing-Gradient-Problem [44] abgeschwächt wird, was das Lernen weiter stabilisiert [68].

3.4.1.6 Ensemble von neuronalen Netzen

Bei Ensembles von neuronalen Netzen [124] werden mehrere gleichartige neuronale Netze mit unterschiedlichen Startinitialisierungen gelernt. Als Lösung wird schließlich der Durchschnittswert der Ausgabewerte angegeben. Diese relativ aufwändige Regularisierungstechnik ist durch die Bayes-Inferenz motiviert und hat sich in der Praxis bewährt. Danach ergibt sich für die Lösung y für ein bestimmtes $\mathbf{x}$ in Abhängigkeit von den bereits gemachten Beobachtungen Z die Wahrscheinlichkeit

$$P(\mathbf{x}, y|Z) \quad = \quad \int_{\omega} P(\mathbf{x}, y|\omega) P(\omega|Z) d\omega \tag{3.51}$$

durch Integrieren über alle möglichen Modelle gewichtet mit deren a priori Wahrscheinlichkeiten. Hierbei wird jedes Modell als erwartungstreuer Schätzer für die Menge aller möglichen Modelle angenommen. Diese an das Bagging [40] angelehnte Technik kann jedoch nicht als Regularisierungsmethode, die mit dem Weight-Decay vergleichbar ist, betrachtet werden. Die unterschiedlichen Modelle basieren auf der Unzulänglichkeit des Gradientenverfahrens, das globale Optimum zu finden und nicht auf unterschiedlichen Daten. Vielmehr wird über die möglichen lokalen Minima gemittelt, so dass die Fehlerfunktion insgesamt als geglättet interpretiert werden kann. So wird die Varianz der Entfernung vom globalen Minimum oder guter lokaler Minima verringert.

3.4.2 Kernel-Verfahren und Support-Vector-Machines

Kernel-Machines sind den neuronalen Netzen paradigmatisch gegenübergestellt. Die Grundidee der meisten Kernel-Verfahren besteht darin, den Eingaberaum so in einen, i.d.R. höherdimensionalen Feature-Raum zu transformieren, dass die vorherrschenden Zusammenhänge linear beschrieben werden können.

Die Support-Vector-Machine (SVM) [116, 117, 17] ist ein weit verbreitetes Kernel-Verfahren, da sie sich die Erkenntnisse der statistischen Lerntheorie zu Nutze macht. Dazu identifiziert die SVM innerhalb des Feature-Raumes schließlich den Maximum-Margin-Vektor, der ein Klassifikations- oder Regressionsproblem löst. Innerhalb des Eingaberaumes entspricht dieser Vektor einer komplexen Funktion.

3.4.2.1 Maximum-Margin-Klassifikation

Im Klassifikationsfall, durch den die SVM ursprünglich motiviert wurde, gehen wir zunächst davon aus, dass keine Transformation in einen Feature-Raum durchgeführt wird. Darüber

hinaus seien die vorliegenden Eingabevektoren linear separierbar. Dann ist der Maximum-Margin-Klassifikator definiert als die Funktion

$$f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} - b\right) \tag{3.52}$$

$$(\mathbf{w}, b) = \arg \min_{y(\mathbf{w}^T\mathbf{x}-b)\geq 1} \|\mathbf{w}\| \tag{3.53}$$

für alle Trainingsdaten $(\mathbf{x}, y)$, denn offensichtlich entspricht die Maximierung des Margins der Minimierung der Norm des Gewichtsvektors bei einem als konstant vorgegebenen Margin. Um dieses Optimierungsproblem zu lösen, bedient man sich des Lagrange-Formalismus. Aus dem primären Optimierungsproblem

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} = \min \tag{3.54}$$

$$\forall i \in \{1, \ldots, l\} : y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq 1. \tag{3.55}$$

ergibt sich das entsprechende Lagrange-Funktional

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i \left(y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right) - 1\right). \tag{3.56}$$

Der Sattelpunkt, in dem L für $\mathbf{w}$ minimal und für α maximal ist, schließlich beschreibt die Lösung obigen Optimierungsproblems. Wie sich zeigt, sind

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i = 0 \tag{3.57}$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0 \tag{3.58}$$

dafür notwendige Kriterien. Aus dem Lagrange-Formalismus ergibt sich außerdem für alle $i \in \{1, \ldots, l\} : \alpha_i \geq 0$. Einsetzen der erhaltenen Zusammenhänge in das Lagrange-Funktional führt schließlich auf das duale Optimierungsproblem, das zum Beispiel mittels quadratischer Programmierung gelöst werden kann, effiziente Algorithmen finden sich z.B. in [74, 52, 55, 61, 62, 96, 63]. Wir erhalten schließlich

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{3.59}$$

und einen angemessenen Bias b, der stets so zu wählen ist, dass der Margin maximal wird [17]. Als die Support-Vektoren bezeichnen wir nun diejenigen Vektoren $\mathbf{x}_i$, für die die Nebenbedingung Gl. 3.55 mit Gleichheit erfüllt wird, also $\alpha_i > 0$. Diese Support-Vektoren liegen, je nachdem, zu welcher Klasse sie gehören, auf einer der beiden Ebenen parallel zur Separationsebene, die den minimalen maximalen Abstand zu dieser haben. Je kleiner die Anzahl an Support-Vektoren, desto höher die Generalisierungsleistung [17]. Im Zusammenhang zu Satz 2 erhält man wie folgt ein sehr wichtiges Resultat, das außerdem das Maximieren des Margins motiviert.

Satz 3. *(Vapnik) [116, 117] Seien* $F = \left\{ f \,\middle|\, f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} - b\right), \mathbf{w}, \mathbf{x} \in X \subset \mathbb{R}^n, b \in \mathbb{R} \right\}$*, X eine Hyperkugel mit Radius R und d die Anzahl der Support-Vektoren von l gegebenen Trainingsbeispielen, wobei*

$$\mathbf{w} = \sum_{i} y_i \alpha_i \mathbf{x}_i \tag{3.60}$$

und

$$y_i\left(\mathbf{w}^T\mathbf{x}_i - b\right) \geq \delta^*. \tag{3.61}$$

Dann ist der erwartete Fehler von f nicht größer als

$$\mathbf{E}\text{err}(f) \quad \leq \quad \mathbf{E}\min\left(\frac{d}{l}, \frac{R^2}{(\delta^*)^2 l}, \frac{n}{l}\right). \tag{3.62}$$

Der reale Fehler, genauer der erwartete Generalisierungsfehler, ist also beschränkt durch die Anzahl der Support-Vektoren, den Margin und die Dimensionalität des Problems.

3.4.2.2 Übergang zu Kernels

Die Mächtigkeit der Support-Vector-Machine kommt nun dadurch zu Stande, dass man nicht auf den euklidischen Raum beschränkt ist. Der sogenannte Kernel-Trick besteht darin, das Skalarprodukt $\mathbf{x}^T\mathbf{z}$ durch eine Funktion

$$K \quad : \quad \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}, \tag{3.63}$$

zu ersetzen, die Mercers Bedingungen erfüllt [17]. Dann kann $K(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{x})^T\Phi(\mathbf{z})$ auch als Skalarprodukt in einem in der Regel höherdimensionalen Feature-Raum aufgefasst werden. Entscheidend dabei ist, dass die Struktur dieses Feature-Raumes nicht im Detail bekannt sein muss, denn weder für die Lösung des Optimierungsproblems noch für die Klassifikation selbst, ist es erforderlich $\Phi(\mathbf{x})$ direkt zu bestimmen.

3.4.2.3 Soft-Margin-Klassifikation

Für den in der Praxis wichtigen Fall, dass die Eingabevektoren nicht linear separiert werden können, muss man dem Umstand, dass nicht alle Vektoren korrekt klassifiziert werden können, Rechnung tragen. Zu diesem Zweck lassen wir moderate Fehler bei der Klassifikation zu. Dazu werden sogenannte Slack-Variablen ξ_i eingeführt, die den Abstand des Vektors $\mathbf{x}_i$ zu einer der beiden Maximum-Margin-Ebenen in Richtung Separationsebene darstellen. Das Ziel besteht darin, $\|\xi\|_a$ zu minimieren neben der Minimierung von $\|\mathbf{w}\|$. In der C2-Soft-Margin-Klassifikation erhalten wir das primäre Optimierungsproblem

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{2}\|\xi\|_2 \quad = \quad \min \tag{3.64}$$

$$\forall i \in \{1, \ldots, l\} : y_i \left(\sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) - b\right) \quad \geq \quad 1 - \xi_i. \tag{3.65}$$

Das daraus abgeleitete duale Optimierungsproblem stimmt exakt mit dem Maximum-Margin-Optimierungsproblem überein. Lediglich der Kernel wird zu

$$\hat{K}(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{i,j}}{C} \tag{3.66}$$

modifiziert, wobei $\delta_{i,j}$ das Kronecker-Symbol ist mit

$$\delta_{i,j} \quad = \quad 1, i = j \tag{3.67}$$

$$\delta_{i,j} \quad = \quad 0, i \neq j \tag{3.68}$$

und C als Regularisierungsparameter aufgefasst werden kann.

3.4.2.4 Support-Vector-Regression

Analog zur Klassifikation können mit Support-Vector-Machines auch Regressionsprobleme gelöst werden. Die SVM soll dabei eine Abbildung

$$f(\mathbf{x}) \quad = \quad \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) - b \tag{3.69}$$

realisieren. Wir bezeichnen die SVM als konsistent mit den Trainingsvektoren, falls sie höchstens um ein $\varepsilon > 0$ Fehler macht, also

$$\forall i : |f(\mathbf{x}_i) - y_i| \quad \leq \quad \varepsilon. \tag{3.70}$$

Für den Fall, dass die Trainingsdaten nicht korrekt ausgewertet werden können, benutzen wir wieder Slack-Variablen. Wir gehen von folgendem primären Optimierungsproblem aus.

$$\mathbf{w}^T\mathbf{w} + C\left(\|\xi\| + \left\|\hat{\xi}\right\|\right) \quad = \quad \min \tag{3.71}$$

$$\forall i \in \{1,\ldots,l\} : \left(\sum_{j=1}^{l} \alpha_j K(\mathbf{x}_j, \mathbf{x}_i) - b\right) - y_i \quad \leq \quad \varepsilon + \xi_i \tag{3.72}$$

$$\forall i \in \{1,\ldots,l\} : y_i - \left(\sum_{j=1}^{l} \alpha_j K(\mathbf{x}_j, \mathbf{x}_i) - b\right) \quad \leq \quad \varepsilon + \hat{\xi}_i \tag{3.73}$$

$$\xi_i \quad \geq \quad 0 \tag{3.74}$$

$$\hat{\xi}_i \quad \geq \quad 0 \tag{3.75}$$

Das entsprechende duale Problem kann effizient mit Methoden, die z.B. in [74, 95] beschrieben sind, gelöst werden.

3.4.2.5 Kernelised-Ridge-Regression

Für den Spezialfall $\varepsilon = 0$, ist Support-Vector-Regression identisch mit der Kernelised-Ridge-Regression

$$\left\| \begin{pmatrix} K + C\mathbf{I} & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ b \end{pmatrix} - \mathbf{y} \right\| \quad = \quad \min, \tag{3.76}$$

die auf die Lösung eines linearen Gleichungssystems führt.

3.4.2.6 Gauß-Prozesse

Das Konzept der Inferenz mit Gauß-Prozessen (Gaussian Processes) [17] ist eng mit Support-Vector-Regression und Ridge-Regression verwandt. Jedoch beruht es auf dem Bayes'schen Paradigma. Zunächst geht man davon aus, dass die zu approximierende Funktion $f : X \to Y$ einem Gauß-Prozess folgt, also die beobachteten Sample einer multivariaten Gaußverteilung mit Kovarianzmatrix Σ gemäß

$$P\{(f(\mathbf{x}_1),\ldots,f(\mathbf{x}_l)) = (y_1,\ldots,y_l)\} \quad = \quad ce^{-\frac{1}{2}\mathbf{y}^T\Sigma^{-1}\mathbf{y}} \tag{3.77}$$

mit geeignetem c folgen, hier für den normierten Fall mit $\mathbf{E}(f(\mathbf{x}_1),\ldots,f(\mathbf{x}_l)) = 0$ angegeben. Die Kovarianzmatrix ist stets positiv definit und hat damit die gleichen Eigenschaften wie die oben beschriebene Kernel-Matrix, so dass wir $K = \Sigma$ identifizieren.

Ferner wird ein Gauß'sches Rauschen der Funktionswerte angenommen, so dass die bedingte Wahrscheinlichkeit der beobachteten Funktionswerte $f(\mathbf{x})$ gegeben die wahren Funktionswerte ebenfalls einer Gaußverteilung

$$P(\mathbf{y}|f(\mathbf{x})) \quad = \quad ce^{-\frac{1}{2}(\mathbf{y}-f(\mathbf{x}))^T\Omega^{-1}(\mathbf{y}-f(\mathbf{x}))} \tag{3.78}$$

folgt, wobei jedoch y_i als unabhängig von $f(\mathbf{x}_j)$ für $i \neq j$ und ein gleichmäßiges Rauschen angenommen wird, demnach ist $\Omega = \sigma^2\mathbf{I}$.

Um nun die Wahrscheinlichkeit für einen bestimmten Funktionswert $f(\mathbf{x})$ in Abhängigkeit von einer neuen Beobachtung abzuleiten, wendet man zunächst die Bayes-Regel an. Wir erhalten

$$\begin{aligned} &P(f(\mathbf{x}), f(\mathbf{x}_1),\ldots,f(\mathbf{x}_l)|\mathbf{y},\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_l) \\ &= \frac{P(\mathbf{y}|f(\mathbf{x}_1),\ldots,f(\mathbf{x}_l))P(f(\mathbf{x}), f(\mathbf{x}_1),\ldots,f(\mathbf{x}_l)|\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_l)}{P(\mathbf{y}|\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_l)}. \end{aligned} \tag{3.79}$$

Da der Nenner nur von den Beobachtungen abhängt und nicht beeinflusst werden kann, genügt es

$$P = P(\mathbf{y}|f(\mathbf{x}_1),\ldots,f(\mathbf{x}_l))P(f(\mathbf{x}),f(\mathbf{x}_1),\ldots,f(\mathbf{x}_l)|\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_l) \tag{3.80}$$

zu betrachten. Als Erwartungswert dieser Verteilung ergibt sich

$$f(\mathbf{x}) = \mathbf{y}^T\left(K+\sigma^2\mathbf{I}\right)^{-1}\mathbf{k}, \tag{3.81}$$

wobei $\mathbf{k}_i = K(\mathbf{x},\mathbf{x}_i)$, was gerade dem Ridge-Regression-Ergebnis entspricht.

3.4.3 Vergleich neuronaler Netze mit Kernel-Machines

Neuronale Netze gehören zu den ersten Machine-Learning-Verfahren, Kernel-Machines hingegen haben sich erst in den letzten Jahren etabliert. In den Kap. 6 und 7 werden die Eigenschaften sowie Vor- und Nachteile beider Klassen von Funktionsapproximatoren für die Rewards-Regression deutlich, in Abschn. 12.2 geben wir konkrete Anwendungsempfehlungen. Grundsätzliche Eigenschaften beider Methodenklassen können wir wie folgt zusammenfassen.

- Neuronale Netze
  - sind globale Funktionsapproximatoren
  - erlauben einen hohen Grad an Flexibilität bei der Gestaltung der Architekturen wegen der Anwendung des Gradientenverfahrens
  - ermöglichen die Berücksichtigung von Vorwissen durch die Gestaltung der Architektur
  - sind flexibel trainierbar und ermöglichen die Anwendung unterschiedlicher Optimalitätskriterien
  - sind an das biologische Vorbild angelehnt
  - erfordern i.d.R. die Lösung eines nicht-konvexen Optimierungsproblems
- Kernel-Machines
  - sind lokale Funktionsapproximatoren
  - setzen das Konzept der strukturellen Risiko-Minimierung direkt um und erlauben fundierte theoretische Aussagen über ihre Generalisierungsfähigkeit
  - ermöglichen die Berücksichtigung von Vorwissen durch die Gestaltung des Kernels
  - orientieren sich an mathematisch-statistisch motivierten Optimalitätskriterien wie etwa die Maximum-Margin-Lösung
  - erfordern i.d.R. die Lösung eines konvexen Optimierungsproblems

Kapitel 4

Informationseffizienz im Reinforcement-Learning

In vielen praktischen und industriellen Problemstellungen liegt nur eine beschränkte Zahl an Beobachtungen vor, die mit einer i.d.R. begrenzt beeinflussbaren Explorationsstrategie gewonnen wurde. Aus diesem Grund besteht ein großer Bedarf an RL-Verfahren, die möglichst informationseffizient arbeiten, um aus gegebenen Trajektorien gute Policies abzuleiten. Dieses Kapitel soll einen Überblick über Ausgangspunkte für das dateneffiziente Lösen von Reinforcement-Learning-Problemen geben.

Die bedeutendste Maßnahme, jede Beobachtung zu nutzen, wird im Rahmen der sogenannten Fitted-Value-Iteration umgesetzt. Einige der entsprechenden Verfahren sind unabhängig und parallel zu den in der vorliegenden Arbeit in Teil II vorgestellten Beiträgen entstanden. Dazu gehört die Kernelisierung [49, 122] der Least-Squares-Policy-Iteration [57] und ähnliche Bayes'sch motivierte Ansätze [27] sowie die Neural-Fitted-Q-Iteration [83], die verwandt sind mit der Kernel-Rewards-Regression [98, 99] und der Neural-Rewards-Regression [101, 100]. Die Verfahren Least-Squares-Policy-Iteration [57], kernel-basiertes Reinforcement-Learning [69] und Neural-Fitted-Q-Iteration [83] werden in diesem Kapitel vorgestellt und dienen als Grundlage für Teil II und z.T. auch für Teil III.

Im Anschluss daran beschreiben wir exemplarisch drei Verfahren, die das Bayes'sche Paradigma mit Reinforcement-Learning verbinden. Diese sind die Interpretation von Bayesian-Reinforcement-Learning als POMDP [25, 76], das Bayesian-Q-Learning [21] und das Gaussian-Process-Temporal-Difference-Learning [26, 27]. Diese Methoden sollen auf Teil III und z.T. auf Teil II vorbereiten. Wir verzichten auf die Darstellung der in [22] vorgestellten Percentile-Optimisation-Methode für MDP, die in ihrer Zielstellung den Beiträgen in Teil III ähnelt, jedoch andere Techniken benutzt.

Außerdem erfolgt eine Darstellung des Standes der Technik zur Approximation dynamischer Systeme mit rekurrenten neuronalen Netzen [125, 124], die in Teil II im Zusammenhang mit der Neural-Rewards-Regression für die Beherrschung von POMDP von Bedeutung sein werden.

Schließlich geben wir einen Überblick über theoretische Ergebnisse, die mit Dateneffizienz zusammenhängen. Wir stellen einige Performance- und Kapazitätsschranken vor [13, 66, 51, 3, 4].

4.1 Nutzung aller Beobachtungen – Fitted-Value-Iteration

Ein wesentlicher erster Schritt zum dateneffizienten Reinforcement-Learning für kontinuierliche Zustände besteht in der Erkenntnis, dass alle Beobachtungen, also alle Transitionen zur Bestimmung der optimalen Policy vollständig zu nutzen sind. Die vorliegenden Verfahren

Monte-Carlo-RL und TD-Learning sind jedoch online arbeitende Methoden und heben sich die Trajektorien nicht auf.

Die Grundidee der Fitted-Value-Iteration besteht in der Kombination von dynamischer Programmierung mit Regression. Dazu wird die Bellman-Iteration

$$(T^\pi Q)(s,a) = \mathbf{E}_{s'}(R(s,a,s') + \gamma Q(s', \pi(s'))) \tag{4.1}$$

mit einem Projektionsoperator Γ kombiniert, der für einen gegebenen Hypothesenraum H und eine Funktion $f : S \times A \to \mathbb{R}$ diejenige Hypothese $h \in H$ identifiziert, die den kleinstmöglichen Abstand zu f hat. Das Abstandsmaß hängt dabei mit den beobachteten Transitionen zusammen, so dass etwa

$$\|f - h\| = \sum_{i=1}^{l} (f(s_i, a_i) - h(s_i, a_i))^2 = \min. \tag{4.2}$$

Die Projektion kann aber auch andere Abstandsmaße nutzen oder einem Operator entsprechen, der die Beobachtungen auf andere Weise nutzt. Value-Iteration besteht dann in der Iteration

$$\begin{aligned} Q^{t+1} &= \Gamma T^\pi Q^t & (4.3)\\ &= \Gamma \left(R + \gamma P \Pi Q^t\right), & (4.4) \end{aligned}$$

wobei $Q^t \in \mathbb{R}^l$ und $R \in \mathbb{R}^l$ Vektoren, $P \in \mathbb{R}^{l \times l}$ eine Matrix und Π ein Operator, der auf Q wirkt, sind. Es ist im Einzelnen

$$\begin{aligned} Q_i &= Q(s_i, a_i) & (4.5)\\ R_i &= R(s_i, a_i, s_i') & (4.6)\\ P_{i,j} &= P(s_j | s_i, a_i) & (4.7)\\ \Pi_i Q &= \sum_{a \in A} \pi(s_i, a) Q(s_i, a). & (4.8) \end{aligned}$$

Im Detail sind auch andere Formen der Value-Iteration möglich. Entscheidend ist, dass der Operator Γ auf eine Funktion Q wirkt und sich dabei die beobachteten Transitionen zu Nutze macht.

Ein entscheidender Schritt besteht nun darin, den Operator P zu eliminieren und die tatsächlich gemachten Beobachtungen als erwartungstreue Schätzer sowohl für die Rewards als auch die Q-Funktion aufzufassen. Dann erhält man die Fitted-Value-Iteration

$$\begin{aligned} Q^{t+1} &= \Gamma T^\pi Q^t & (4.9)\\ &= \Gamma \left(R + \gamma \Pi Q^t\right). & (4.10) \end{aligned}$$

Diese Iteration konvergiert nach dem Banachschen Fixpunktsatz [8], falls Γ eine Nicht-Expansion ist [13, 57], denn

$$\begin{aligned} \|\Gamma T^\pi Q_1 - \Gamma T^\pi Q_2\| &\leq \|\Gamma\| \|T^\pi Q_1 - T^\pi Q_2\| & (4.11)\\ &\leq \|T^\pi Q_1 - T^\pi Q_2\| & (4.12)\\ &= \|(R + \gamma \Pi Q_1) - (R + \gamma \Pi Q_2)\| & (4.13)\\ &= \|\gamma \Pi (Q_1 - Q_2)\| & (4.14)\\ &\leq \gamma \|\Pi\| \|Q_1 - Q_2\| & (4.15)\\ &\leq \|Q_1 - Q_2\|. & (4.16) \end{aligned}$$

Dies hängt jedoch von der Wahl des Funktionsapproximators und des Hypothesenraumes ab. Für den Optimalitätsoperator T gelten entsprechende Konvergenzaussagen [12]. In diesem Fall ist Π so zu wählen, dass der bzgl. der Aktionen lokal maximale Q-Wert herangezogen wird. Der Einfachheit halber verzichten wir im Folgenden auf Π.

4.1.1 Least-Squares-Policy-Iteration

Bei der Least-Squares-Policy-Iteration (LSPI) [57], vorgestellt von Lagoudakis und Parr, wird H als eine Menge linearer Funktionen innerhalb eines geeigneten Feature-Raumes aufgefasst. Der Feature-Raum der Dimension d entsteht durch eine Transformationsfunktion $\Phi : S \times A \to \mathbb{R}^d$. Dabei bezeichne $\Phi \in \mathbb{R}^{l \times d}$ die Features der beobachteten Zustands-Aktions-Paare (s_i, a_i) und $\Phi' \in \mathbb{R}^{l \times d}$ die der Folgezustände zusammen mit den Folgeaktionen, so dass

$$\Phi_{i,j} = \Phi(s_i, a_i)_j \tag{4.17}$$

$$\Phi'_{i,j} = \sum_{a \in A} \pi(s'_i, a)\Phi(s'_i, a)_j. \tag{4.18}$$

Die Funktionen $f = f_{\mathbf{w}} \in H$ sind vollständig beschrieben durch einen Vektor $\mathbf{w}$, so dass

$$f(s, a) = \Phi(s, a)^T \mathbf{w}. \tag{4.19}$$

Die Fitted-Value-Iteration erhält nun die Gestalt

$$\Phi \mathbf{w}^{t+1} = \Gamma(R + \gamma \Phi' \mathbf{w}^t), \tag{4.20}$$

wobei Γ so gewählt wird, dass

$$\|\Phi \mathbf{w}^{t+1} - (R + \gamma \Phi' \mathbf{w}^t)\| = \min \tag{4.21}$$

bzgl. $\mathbf{w}^{t+1}$ minimiert wird. Dabei findet die euklidische Norm Anwendung, was, zusammen mit der linearen Architektur, den Namen des Verfahrens erklärt. Dieses Optimierungsproblem ist konvex und hat eine eindeutige Lösung, die sich analytisch bestimmen lässt. Sie ergibt sich aus der Lösung des linearen Gleichungssystems

$$\Phi^T \Phi \mathbf{w}^{t+1} = \Phi^T (R + \gamma \Phi' \mathbf{w}^t). \tag{4.22}$$

Ein Fixpunkt für $\mathbf{w}$ ist dabei erreicht, wenn

$$\Phi^T \Phi \mathbf{w} = \Phi^T (R + \gamma \Phi' \mathbf{w}) \tag{4.23}$$

$$\Leftrightarrow \Phi^T (\Phi - \gamma \Phi') \mathbf{w} = \Phi^T R \tag{4.24}$$

$$\Leftrightarrow \mathbf{w} = \left(\Phi^T (\Phi - \gamma \Phi')\right)^{-1} \Phi^T R. \tag{4.25}$$

Policy-Evaluation kann daher im Wesentlichen mit einer Matrix-Inversion gelöst werden. Im Policy-Iteration-Fall ergeben sich die Alternativen der Fitted-Value-Iteration wie oben beschrieben oder der Policy-Iteration durch sukzessive Policy-Evaluation und Anpassen der Policy. Im Gegensatz zu den Unterschieden für den diskreten Fall in Abschn. 2.2.2 ergibt sich daher hier kein asymptotischer Komplexitätsunterschied der beiden Methoden. Die Konvergenz von Policy-Iteration kann nur garantiert werden, falls die Spektralnorm des Projektionsoperators

$$\left\| \Phi' \left(\Phi^T \Phi\right)^{-1} \Phi^T \right\| < 1. \tag{4.26}$$

4.1.2 Kernel-basiertes Reinforcement-Learning

Die Arbeit von Ormoneit und Sen zum kernel-basierten Reinforcement-Learning [69] ist eine der ersten Ansätze zum Einsatz von Kernels im RL. Wie bei den meisten Kernel-Verfahren beruht der Funktionsapproximator auf Stützstellen (Supporting-Points) und einem geeignet gewählten Kernel. Die Approximation selbst erfolgt durch Kernel-Glättung, was einem geglätteten Mittelwert entspricht.

Die Q-Funktionen werden dargestellt als

$$Q(s,a) = \sum_{i=1}^{n} Q_i^a K(s,s_i), \tag{4.27}$$

wobei die Q_i^a zu bestimmende Skalare sind. Die s_i sind die n Stützstellen, ausgewählte bereits beobachtete Zustände. Dabei kann für jede Aktion a eine eigene Menge an Stützstellen bestimmt werden. Der Projektionsoperator Γ ist für jede Aktion separat zu bestimmen und ergibt sich zu

$$(\Gamma_a f)(s,a) = \sum_{j=1}^{n} (R(s_j,a,s'_j) + \gamma f(s'_j, \pi(s'_j)))K(s,s_j) \tag{4.28}$$

für Policy-Evaluation oder

$$(\Gamma_a f)(s,a) = \sum_{j=1}^{n} (R(s_j,a,s'_j) + \gamma \max_{a'} f(s'_j,a'))K(s,s_j) \tag{4.29}$$

für Policy-Iteration. Ormoneit und Sen schlagen die Anwendung abstandsbasierter und normierter Kernels

$$K(s,s_i) = \frac{\Phi\left(\frac{\|s-s_i\|}{b}\right)}{\sum_{j=1}^{n} \Phi\left(\frac{\|s-s_j\|}{b}\right)} \tag{4.30}$$

vor. Der Kernel muss bewirken, dass Gl. 4.29 tatsächlich eine gewichtete Mittelung darstellt. Der Fixpunkt dieser Iteration ist erreicht, falls

$$\forall i : Q(s_i,a_i) = \sum_{j=1}^{n} \left(R(s_j,a,s'_j) + \gamma \max_{a'} Q(s'_j,a')\right) K(s_i,s_j). \tag{4.31}$$

Ormoneit und Sen haben gezeigt, dass auch im Policy-Iteration-Fall stets Konvergenz garantiert werden kann und die entstehende Q-Funktion beliebig genau die wahre Q-Funktion annähert, sofern die Anzahl der beobachteten Transitionen gegen unendlich geht.

4.1.3 Neural-Fitted-Q-Iteration

Die Neural-Fitted-Q-Iteration (NFQ) [83] von Riedmiller ist Value-Iteration (oder Q-Iteration) mit neuronalen Netzen. NFQ ist, im Vergleich zu den oben beschriebenen Ansätzen, die am engsten mit dem TD-Learning verwandte Methode. Als Funktionsapproximator werden hier neuronale Netze verwendet, die mit geeigneten Back-Propagation-Verfahren trainiert werden. Riedmiller verwendet selbst das Resilient-Propagation-Verfahren [84]. Er hat die NFQ als einer der ersten explizit mit dem Ziel, dateneffizient RL-Probleme zu lösen, entwickelt und erzielte auf wichtigen Benchmark-Problemen, die auch von Lagoudakis und Parr untersucht wurden [57], vergleichbare Ergebnisse mit weniger beobachteten Transitionen.

Die Q-Funktion wird dargestellt als ein neuronales Netz $Q_\theta(s,a)$ mit Parametern θ. Die Projektion Γ erfolgt in jeder Iteration dann durch einen vollständigen Gradientenabstieg in Richtung

$$\begin{aligned}\Delta\theta^{t+1} = & -\alpha \sum_{i=1}^{l} (Q_{\theta^{t+1}}(s_i,a_i) - R(s_i,a_i,s'_i) - \gamma Q_{\theta^t}(s'_i,\pi(s'_i))) \\ & \frac{\partial}{\partial\theta^{t+1}} Q_{\theta^{t+1}}(s_i,a_i)\end{aligned} \tag{4.32}$$

für Policy-Evaluation oder

$$\begin{aligned}\Delta\theta^{t+1} &= -\alpha \sum_{i=1}^{l} \left(Q_{\theta^{t+1}}(s_i, a_i) - R(s_i, a_i, s'_i) - \gamma \max_{a'} Q_{\theta^t}(s'_i, a') \right) \\ &\quad \frac{\partial}{\partial \theta^{t+1}} Q_{\theta^{t+1}}(s_i, a_i) \end{aligned} \tag{4.33}$$

für Policy-Iteration. Ein Fixpunkt $Q = \Gamma Q$ ist in diesem Fall dann erreicht, wenn Q das Ergebnis des Gradientenabstiegs unter eigener Anwendung ist, also wenn

$$\sum_{i=1}^{l} (Q_\theta(s_i, a_i) - R(s_i, a_i, s'_i) - \gamma \max_{a'} Q_\theta(s'_i, a')) \frac{\partial}{\partial \theta} Q_\theta(s_i, a_i) = 0. \tag{4.34}$$

Problematisch sind hier zweierlei Effekte. Zum Einen können in jedem Iterationsschritt lokale Minima erreicht werden, die dann als Schätzer für die Q-Funktion in der nächsten Iteration herangezogen werden. Und zum Anderen ist der Projektionsoperator Γ im Allgemeinen keine Nicht-Expansion, so dass die Iteration nicht konvergieren muss, auch wenn in jedem Iterationsschritt das globale Minimum gefunden wird. Beide Effekte schlagen sich nieder im sogenannten Policy-Degradation-Phänomen [29], das dazu führt, dass gut gelernte Q-Funktionen nach weiteren Iterationen wieder an Qualität verlieren können. Darüber hinaus kann auch die Laufzeit dieses Verfahrens für die Praxis problematisch sein, da iterativ mehrere Gradientenabstiegsverfahren vollständig ausgeführt werden müssen, was im Vergleich zu den Iterationen der in den Abschn. 4.1.1 und 4.1.2 beschriebenen Verfahren wesentlich zeitaufwändiger ist.

Dennoch hat die NFQ die bisher beste Informationseffizienz erzielt, da das Prinzip der vollständigen Datennutzung mit neuronalen Netzen als mächtigen Funktionsapproximatoren kombiniert wird.

4.2 Auxiliared-Bellman-Residuum

Die im vorigen Abschnitt vorgestellten Verfahren streben die sogenannte Fixpunkt-Lösung an, die der Lösung des TD-Learning entspricht. Bei der Rewards-Regression (siehe Teil. II) spielt neben diesem Optimalitätskriterium das Minimum des Bellman-Residuums eine wichtige Rolle, das wir bereits in Abschn. 2.5.2 eingeführt haben. In diesem Abschnitt weisen wir auf die Einschränkungen des Bellman-Residuums für Reinforcement-Learning im Batch-Learning hin und stellen einen Ansatz zur begrenzten Bewältigung dieser Einschränkungen vor [3, 4].

Das Minimieren des Bellman-Residuums hat auf der einen Seite den Vorteil, ein kontrollierbares Lernproblem zu sein, da es eng mit Supervised-Learning-Problemstellungen verwandt ist. Auf der anderen Seite tendiert es dazu, Terme von Momenten höherer Ordnung des erwarteten Gesamtertrags im stochastischen Fall zu minimieren, falls keine weiteren unkorrelierten Beobachtungen für vom gleichen Zustand ausgehende Transitionen zur Verfügung stehen [107, 57]. Im Allgemeinen sind die Lösungen zu Gunsten einer stärkeren Regularisierung der Q-Funktionen für Folgezustände stark stochastischer Transitionen verschoben, da die Varianz des Gesamtertrags als Fehlerterm berücksichtigt werden muss. Die angestrebte Q-Funktion hat daher einen größeren Bias in Richtung einfacher Funktionen mit geringen Schwankungen, insbesondere an Stellen, die alternativen Folgezuständen entsprechen.

Dies kann man wie folgt einsehen. Falls s'_i und r_i erwartungstreue Schätzer der Folgezustände und Rewards sind, dann ist der Ausdruck $(Q(s_i, a_i) - \gamma V(s'_i) - r_i)^2$ kein erwartungstreuer Schätzer für das wahre quadratische Bellman-Residuum $(Q(s, a) - (TQ)(s, a))^2$, sondern für $(Q(s, a) - (TQ)(s, a))^2 + (T'Q)(s, a)^2$, da

$$B(s, a) = \mathbf{E}_{s'}(Q(s, a) - (\gamma V(s') + R(s, a, s')))^2 \tag{4.35}$$

$$
\begin{aligned}
&= \mathbf{E}_{s'}\big(Q(s,a)^2 - 2Q(s,a)(\gamma V(s') + R(s,a,s')) \\
&\quad +(\gamma V(s') + R(s,a,s'))^2\big) && (4.36)\\
&= \big(Q(s,a)^2 - 2Q(s,a)\mathbf{E}_{s'}(\gamma V(s') + R(s,a,s')) \\
&\quad +\mathbf{E}_{s'}(\gamma V(s') + R(s,a,s'))^2\big) && (4.37)\\
&= \big(Q(s,a)^2 - 2Q(s,a)\mathbf{E}_{s'}(\gamma V(s') + R(s,a,s')) \\
&\quad +(\mathbf{E}_{s'}(\gamma V(s') + R(s,a,s')))^2 + \mathbf{var}_{s'}(\gamma V(s') + R(s,a,s'))\big) && (4.38)\\
&= (Q(s,a) - \mathbf{E}_{s'}(\gamma V(s') + R(s,a,s')))^2 \\
&\quad +\mathbf{var}_{s'}(\gamma V(s') + R(s,a,s')) && (4.39)\\
&= (Q(s,a) - (TQ)(s,a))^2 + (T'Q)(s,a)^2. && (4.40)
\end{aligned}
$$

Als eine Alternative für das Nutzen doppelter Transitionen (s,a,s'_1) und (s,a,s'_2), bei deren Anwendung sich

$$
\begin{aligned}
B'(s,a) &= \mathbf{E}_{(s'_1,s'_2)}((Q(s,a) - (\gamma V(s'_1) + R(s,a,s'_1))) \\
&\quad (Q(s,a) - (\gamma V(s'_2) + R(s,a,s'_2)))) && (4.41)\\
&= Q(s,a)^2 - Q(s,a)\mathbf{E}_{(s'_1,s'_2)}(((\gamma V(s'_1) + R(s,a,s'_1)) \\
&\quad +(\gamma V(s'_2) + R(s,a,s'_2)))) \\
&\quad +\mathbf{E}_{(s'_1,s'_2)}((\gamma V(s'_1) + R(s,a,s'_1))(\gamma V(s'_2) + R(s,a,s'_2))) && (4.42)\\
&= Q(s,a)^2 - 2Q(s,a)\mathbf{E}_{s'}(\gamma V(s') + R(s,a,s')) \\
&\quad + (\mathbf{E}_{s'}(\gamma V(s'_1) + R(s,a,s')))^2 && (4.43)\\
&= (Q(s,a) - \mathbf{E}_{s'}(\gamma V(s') + R(s,a,s')))^2 && (4.44)\\
&= (Q(s,a) - (TQ)(s,a))^2, && (4.45)
\end{aligned}
$$

ergibt, was keine Option im Batch-Learning-Fall ist, schlagen Antos et al. die Anwendung eines modifizierten Bellman-Residuums als besseren Schätzer an das wahre Bellman-Residuum vor [3, 4], das wir als unterstütztes Bellman-Residuum (Auxiliared-Bellman-Residual) bezeichnen. Es ist definiert als

$$
\begin{aligned}
L_{\text{aux}} &= L - \hat{L} && (4.46)\\
&= \sum_{i=1}^{l} (Q(s_i,a_i) - \gamma V(s'_i) - r_i)^2 - \sum_{i=1}^{l} (h(s_i,a_i) - \gamma V(s'_i) - r_i)^2. && (4.47)
\end{aligned}
$$

Die Aufgabe besteht dann im Lösen von

$$
\hat{Q} = \arg \min_{Q \in H_Q} \max_{h \in H_h} L_{\text{aux}}. \qquad (4.48)
$$

Die zu Grunde liegende Idee besteht darin, ein h zu finden, das den Bellman-Operator über die Beobachtungen möglichst gut approximiert. Dieser unvermeidbare Fehler entspricht dann gerade der Varianz im ursprünglichen Ausdruck. Diese Technik erlaubt, das wahre Bellman-Residuum nach oben zu beschränken, falls der Fehler von h für die Schätzung von TQ beschränkt werden kann [3, 4]. Wir werden das Auxiliared-Bellman-Residuum in den Abschn. 5.4 und 7.3 wieder aufgreifen.

4.3 Bayesian-Reinforcement-Learning

Wie wir bereits gesehen haben, ist Bayesian-Learning ein konstruktives Vorgehen zum Schätzen von Parametern und Ergebnissen etwa von Klassifikations- oder Regressionsproblemen (siehe Abschn. 3.3).

Im Reinforcement-Learning gibt es eine Vielzahl von Möglichkeiten, die Bayes'sche Perspektive einzusetzen. Grundsätzlich unterscheidet man zwischen modellbasiertem und modellfreiem Bayesian-RL. Beim modellbasierten Bayesian-RL wird ein Prior über die Modell-Parameter, also die Eigenschaften des MDP definiert, während beim modellfreien Bayesian-RL der Prior direkt über die Bewertungsfunktion oder Q-Funktion festgelegt wird. Im Folgenden sollen drei grundsätzliche Ansätze Bayes'schen Reinforcement-Learning skizziert werden.

4.3.1 Bayesian-Reinforcement-Learning als POMDP

Der Ansatz, Bayesian-RL als POMDP aufzufassen, wurde von Duff in [25] untersucht und angewendet und weiter von Poupart et al. etwa in [76] analysiert. Das Modell geht von einem diskreten Zustandsraum aus. Ferner seien die Rewards als bekannt angenommen und die Transitionen einer unbekannten Verteilung P_T unterworfen, die mit Parametern $\theta_{s,a,s'}$ beschrieben werden können. In [76] wird argumentiert, dass durch das künstliche Einführen von Transitionen zur Ermittlung der Rewards obige Einschränkung aufgehoben wird, dazu muss jedoch die Reward-Funktion hinreichend fein diskretisiert werden.

Die Überführung in partiell beobachtbare Probleme geschieht durch die Definitionen

$$\begin{aligned} S_{\text{POMDP}} &= S_{\text{MDP}} \times \{\theta_{s,a,s'}\} && (4.49)\\ U &= S_{\text{MDP}} && (4.50)\\ P(s'|s,\theta_{s,a,s'},a) &= \theta_{s,a,s'} && (4.51)\\ P(\theta'_{s,a,s'}|\theta_{s,a,s'}) &= \delta_{\theta_{s,a,s'},\theta'_{s,a,s'}}. && (4.52) \end{aligned}$$

Der Zustandsraum des POMDP besteht also aus dem Zustandsraum des MDP und der unbekannten Parameter der Transitionswahrscheinlichkeiten, während die beobachtbaren Größen aus den Zuständen des ursprünglichen MDP bestehen. Dann kann eine in POMDP als Belief-Monitoring bezeichnete Vorgehensweise zur Anwendung kommen, bei der nach jeder beobachteten Transition der Belief $b(\theta)$, der eine Verteilung über θ darstellt, mit Hilfe der Bayes-Regel gemäß

$$b_{\theta,\text{neu}} = cb_\theta P(s'|\theta,s,a) \quad (4.53)$$

aktualisiert wird. Dabei ist $P(s'|\theta,s,a)$ der Likelihood, da s und a bereits in der Vorgängertransition beobachtet und in b_θ integriert wurden, b_θ muss initial als Prior gesetzt werden. Unter der Annahme multinomial verteilter Transitionen für Zustands-Aktions-Paare zeigt sich, dass die Dirichlet-Verteilung als konjugierter Prior angewendet werden kann, die dann b_θ zu Grunde liegt.

Die zu bestimmende Bewertungsfunktion wird schließlich abhängig gemacht vom Belief b und es ergibt sich die Bellman-Optimalitäts-Gleichung

$$V^*(s,b) = \max_a \sum_{s'} P(s'|s,b,a)(R(s,a,s') + \gamma V^*(s',b)). \quad (4.54)$$

Poupart et al. nutzen schließlich Ergebnisse über die Form der Bewertungsfunktion im POMDP-Fall [104] und spezifizieren diese für das Bayesian-RL [76], die wiederum genutzt werden können, um einen effizienten Algorithmus zu entwerfen.

4.3.2 Bayesian-Q-Learning

Bayesian-Q-Learning [21] basiert auf dem Q-Learning-Algorithmus und ist eine der ersten Arbeiten zum Bayesian-RL. Dearden et al. haben selbst später auf diese Arbeit aufgebaut und das Ziel einer informationseffizienten gerichteten Exploration [20] noch stärker forciert.

Hier wird ebenfalls von diskreten Zuständen ausgegangen, jedoch eine Unsicherheit der Q-Funktion selbst angenommen, also ein Prior über die möglichen Q-Werte definiert. Dazu

wird angenommen, dass der Gesamtertrag durch eine normal-verteilte Zufallsvariable

$$D(s,a) = \sum_{t=0}^{\infty} \gamma^t R\left(s^{(t)}, \pi\left(s^{(t)}\right), s^{(t+1)}\right) \tag{4.55}$$

beschrieben werden kann, wobei alle $D(s,a)$ unabhängig voneinander für verschiedene Zustands-Aktions-Paare sind. Der Erwartungswert $\mathbf{E}D(s,a)$ ist dann ein Schätzer für $Q(s,a)$. Als Prior über die Parameter μ und σ^2, die Erwartungswert und Varianz beschreiben, wird eine Normal-Gamma-Verteilung angenommen, die ein konjugierter Prior für die Normalverteilung ist.

Nach dem Beobachten von Transitionen erfolgt unter der Annahme deterministischer Rewards r die Aktualisierung von Mittelwert und Varianz durch

$$\mu_{\text{neu}} = \mathbf{E}(r + \gamma D(s,a)) \tag{4.56}$$
$$= r + \gamma\mu \tag{4.57}$$
$$\sigma^2_{\text{neu}} = \mathbf{E}(r + \gamma D(s,a))^2 - \mu^2_{\text{neu}} \tag{4.58}$$
$$= r^2 + 2\gamma r\mu + \gamma^2\left(\sigma + \mu^2\right) - \mu^2_{\text{neu}}. \tag{4.59}$$

Diese Parameter werden schließlich benutzt, um Daten zu generieren, die dann zur Aktualisierung der tatsächlichen a posteriori Verteilung verwendet werden. Die Beschränkung auf deterministische Rewards führt jedoch zu einem schnellen Verschwinden der Varianz. Dieses Problem wird in [21] durch verschiedene Heuristiken zu beheben versucht. Das Problem liegt letztlich in der Unterscheidung von intrinsischer und extrinsischer Stochastizität [60], die in Abschn. 9.2 noch im Detail diskutiert wird.

Das zentrale Ziel der Arbeit, eine gerichtete Exploration zu realisieren, wird schließlich erreicht, indem Aktionen a mit der Wahrscheinlichkeit $P\{Q(s,a) > Q(s,a')\forall a' \neq a\}$ gesampelt werden, wobei P die aktuell geschätzte Verteilung ist.

4.3.3 Gaussian-Process-Temporal-Difference-Learning

Die Idee, Gauß-Prozesse für Policy-Evaluation und Policy-Iteration einzusetzen, wurde erstmals von Engel et al. in [26, 27] und unabhängig davon von Rasmussen und Kuss in [80] verfolgt. Wir wollen hier exemplarisch das Gaussian-Process-TD-Learning (GPTD-Learning) [26, 27] vorstellen, da dazu eine Reihe von Folgearbeiten vorliegen, in denen die Techniken mit Policy-Gradient- und Actor-Critic-Ansätzen kombiniert werden [33, 34].

Beim GPTD-Learning wird ein Prior über die Bewertungsfunktion in Form eines normierten Gauß-Prozesses angenommen, also

$$\mathbf{E}(V(s)) = 0 \tag{4.60}$$
$$\mathbf{E}(V(s)V(\hat{s})) = K(s,\hat{s}). \tag{4.61}$$

Die Kovarianz zwischen zwei Beobachtungen kann dabei in Form eines Kernels K ausgedrückt werden.

Beim GPTD-Learning für Policy-Evaluation wird nun der Reward-Prozess als

$$R(s,s') = V(s) - \gamma V(s') + N(s) \tag{4.62}$$

modelliert, wobei N eine normalverteilte Zufallsvariable mit Erwartungswert 0 und Kovarianz $\sigma(s)\delta(s-\hat{s})$ beschreibt, es ergibt sich also eine Diagonalmatrix Σ mit Einträgen σ als Kovarianzmatrix für den Rausch-Prozess. Der Prior über den Reward-Prozess, der durch K und Σ beschrieben wird, führt nach l aufeinanderfolgenden Beobachtungen, wie in [26] ausgeführt wird, schließlich zu einer a posteriori Gaußverteilung für den Wert der Bewertungsfunktion für eine neue Beobachtung, die mit

$$\mathbf{E}(V(s)) = \mathbf{k}^T H^T\left(HKH^T + \Sigma\right)^{-1}\mathbf{r} \tag{4.63}$$
$$\mathbf{var}(V(s)) = K(s,s) - \mathbf{k}^T H^T\left(HKH^T + \Sigma\right)^{-1} H\mathbf{k} \tag{4.64}$$

bestimmt wird. Dabei sind $\forall i \in \{1,\ldots,l\} : \mathbf{k}_i = K(s,s_i)$, $\mathbf{r}$ der Vektor mit den ersten $l-1$ Rewards und $H \in \mathbb{R}^{(l-1)\times l}$ eine Matrix, deren Einträge 0 sind mit Ausnahme von $H_{i,i} = 1$ und $H_{i,i+1} = -\gamma$ jeweils $\forall i \in \{1,\ldots,l-1\}$.

Diese Methode ist nur dann korrekt, wenn der zu Grunde liegende MDP deterministisch ist mit unabhängigem Gauß-Rauschen auf den Rewards. In [27] beheben Engel et al. diese Einschränkung und erweitern die Methode darüber hinaus für Policy-Iteration. Dazu wird der Gesamtertrag als

$$\begin{aligned} D(s) &= R(s) + \gamma D(s') && (4.65)\\ &= V(s) + \Delta V(s) && (4.66) \end{aligned}$$

modelliert. Das oben beschriebene Reward-Modell wird so modifiziert, dass

$$\begin{aligned} R(s,s') &= V(s) - \gamma V(s') + N(s,s') && (4.67)\\ &= V(s) - \gamma V(s') + (\Delta V(s) - \gamma \Delta V(s')), && (4.68) \end{aligned}$$

wobei die $\Delta V(s)$ für verschiedene Zustände als unabhängig voneinander betrachtet werden. Entscheidend ist, dass D einer intrinsischen Stochastizität unterliegt, die sich von der extrinsischen Stochastizität, also der Unsicherheit, die es zu modellieren gilt, unterscheidet. Die Modellierung der Bewertungsfunktion selbst erfolgt wieder wie oben beschrieben, jedoch wird nun N, genauer ΔV ebenfalls als Gauß-Prozess modelliert. Da sich auf Grund der Annahmen $\mathbf{E}(\Delta V(s)) = 0$ und $\mathbf{E}(\Delta V(s)\Delta V(\hat{s})) = 0$ ergeben, bleibt $\mathbf{E}(\Delta V(s)^2) = \mathbf{var}(D(s))$, das wir zu $\mathbf{var}(D(s_i)) = \sigma_i^2$ definieren. Unter der Annahme, dass $\sigma_i = \sigma \forall i$ ergibt sich

$$\Sigma = \sigma^2 \begin{pmatrix} 1+\gamma^2 & -\gamma & 0 & \cdots & 0 \\ -\gamma & 1+\gamma^2 & -\gamma & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \cdots & -\gamma & 1+\gamma^2 \end{pmatrix}. \quad (4.69)$$

Die a posteriori Verteilung der Bewertungsfunktion ergibt sich dann wie in den Gln. 4.63 und 4.64. Als Policy-Iteration-Methode haben Engel et al. schließlich einen an SARSA (siehe Abschn. 2.4) angelehnten GPSARSA-Algorithmus entwickelt.

Die hier vorgestellten Verfahren verwenden bereits Bayes'sche Methoden zur Bestimmung der Policy, stellen jedoch unterschiedliche Anforderungen an die Rewards, die verwendeten Prior oder geben nur begrenzte Möglichkeiten, die Prior zu begründen, etwa wenn sie sich direkt auf die Bewertungsfunktion beziehen. In Kap. 9 werden wir ein Verfahren vorstellen, das diese Einschränkungen überwindet und in einem breiten Kontext angewendet werden kann.

4.4 Dynamische Systeme und rekurrente neuronale Netze

Die Modellierung dynamischer Systeme spielt zur Ausgestaltung von RL-Verfahren eine besondere Rolle für modellbasiertes RL und POMDP. In Kap. 7 werden wir genau für diese Fälle rekurrente Netze zu ihrer Modellierung anwenden.

4.4.1 Dynamische Systeme

Dynamische Systeme [82] sind vollständig definiert durch einen Phasenraum M und eine Übergangsfunktion $\phi^t : M \to M$ mit Parameter t, der die Semantik der Zeit annimmt. Dabei betrachten wir hier ausschließlich zeitdiskrete Systeme, für die also $t \in \mathbb{N}$. Dynamische Systeme haben insbesondere die Eigenschaft, dass sie transitiv in t sind, also

$$\phi^{t_1}\left(\phi^{t_2}(\mathbf{x})\right) = \phi^{t_1+t_2}(\mathbf{x}). \quad (4.70)$$

Offensichtlich sind dynamische Systeme also eine Teilmenge von Markov-Prozessen mit einer deterministischen Transitionsübergangswahrscheinlichkeit P_T.

Man unterscheidet ferner geschlossene und offene dynamische Systeme. Bei geschlossenen Systemen wird der Zustand $\mathbf{x}$ des Systems ausschließlich durch ϕ^t beeinflusst. Offene Systeme hingegen können von der Umwelt beeinflusst werden. In offenen dynamischen Systemen wird der Zustand $\mathbf{x}$ zum Zeitpunkt t mittels

$$\mathbf{x}_t \quad = \quad g(\mathbf{x}_{t-1}, s_t) \tag{4.71}$$

bestimmt. Damit aus der Modellierung solcher Systeme ein Nutzen gezogen werden kann, definiert man zusätzlich eine Ausgabefunktion

$$y_{t+1} \quad = \quad h(\mathbf{x}_t), \tag{4.72}$$

die eine funktionale Abhängigkeit vom internen Zustand des Systems darstellt. Diese kann genutzt werden für typische Machine-Learning-Aufgaben wie Klassifikation, Regression oder Prognose. Im letzteren Anwendungsfall geht es zumeist um das Identifizieren einer Abbildung

$$h(\mathbf{x}_t) \quad \rightarrow \quad s_{t+1}, \tag{4.73}$$

also um die Vorhersage der Eingabegrößen (oder einer Teilmenge der Eingaben) zum nächsten Zeitpunkt. Dies ist ein typisches Vorgehen etwa zur Identifikation mehrdimensionaler Zeitreihen. Im Unterschied zu den in Abschn. 3.4 beschriebenen Methoden ist hier jedoch eine dynamische Abbildung möglich, in der die Vergangenheit der Eingabegrößen mit einbezogen werden kann. Damit dies funktioniert, muss der interne Zustand $\mathbf{x}$ das dynamische System zusammen mit g vollständig beschreiben. Haykin schreibt dazu [41]:

> [Der Zustand beschreibt] die Menge aller Informationen über das Verhalten des Systems in der Vergangenheit, abgesehen von den externen Eingaben.

Eine spezielle Ausprägung stellen dynamische Systeme mit teilweise bekannter Umgebung dar. In dem Fall, dass y_t nicht perfekt approximiert werden kann, geht man von der Annahme aus, dass dies der unvollständigen Beschreibung des dynamischen Systems zur Last gelegt werden kann und man ersetzt

$$\mathbf{x}_t \quad = \quad g\left(\mathbf{x}_{t-1}, s_t, y_t - h\left(\mathbf{x}_{t-1}\right)\right), \tag{4.74}$$

wobei $h(\mathbf{x}_{t-1})$ den tatsächlichen Ausgang des Systems zum Zeitpunkt $t-1$ darstellt [126]. Damit können unerwartete Entwicklungen des Systems besser berücksichtigt werden.

4.4.2 Rekurrente neuronale Netze

Rekurrente neuronale Netze (RNN) sind ein Modell, offene dynamische Systeme abzubilden. Auf Grund ihres Aufbaus sind sie, im Vergleich zu statischen Feed-Forward-Netzen (siehe Abschn. 3.4), geeignet, die intrinsische zeitliche Entwicklung solcher Systeme zu beschreiben.

4.4.2.1 Das Zustandsraum-Modell

Das Zustandsraum-Modell definiert die Übertragung der Eigenschaften dynamischer Systeme in die Netzwerkstruktur. Dabei wird der interne Zustand $\mathbf{x}_t$ des Systems durch die Aktivierung, also das Ergebnis der Aktivierungsfunktion, eines versteckten Clusters des Netzes dargestellt, der aus der notwendigen Anzahl an Neuronen besteht. Die Eingaben aus dem internen Vorgängerzustand $\mathbf{x}_{t-1}$ und den Umwelteinflüssen s_t werden dann mittels

$$\mathbf{x}_t \quad = \quad f(A\mathbf{x}_{t-1} + Bs_t) \tag{4.75}$$

berechnet, wobei f die Übergangsfunktion ϕ repräsentiert und gleichzeitig die Aktivierungsfunktion des versteckten Cluster ist sowie A und B geeignete Konnektoren sind. Der Ausgabewert wird durch

$$y_{t+1} \quad = \quad C\mathbf{x}_t \tag{4.76}$$

mit Hilfe eines Konnektors C bestimmt. Natürlich sind auch komplexere Funktionen für die Gln. 4.75 und 4.76 möglich, jedoch ist diese Definition theoretisch ausreichend zur Abbildung beliebiger dynamischer Systeme [91], falls die Anzahl der Neuronen innerhalb des Clusters für $\mathbf{x}$ hinreichend groß gewählt wird. Der Beweis erfolgt in Anlehnung an [46].

4.4.2.2 Zeitliche Entfaltung

Diese intrinsische rekurrente Struktur wird nun so in ein statisches Problem überführt, dass das zeitinvariante Verhalten des Systems abgebildet wird (siehe Abb. 4.1). Man bedient sich dabei einer als zeitlichen Entfaltung bezeichneten Vorgehensweise. Dabei wird das rekurrente Netz in ein aus $\tau + 1$ versteckten Schichten bestehendes Feed-Forward-Netz überführt, die den internen Zuständen $\mathbf{x}_{t-\tau}$ bis $\mathbf{x}_t$ entsprechen. Diese sind jeweils mit einem Eingabecluster für die Umweltgrößen $s_{t-\tau}$ bis s_t und einem Ausgabecluster für die Ausgaben $y_{t-\tau+1}$ bis y_{t+1} verbunden. Ist das dynamische System mit endlichem Zeithorizont τ beschreibbar, dann ist diese Abbildung eindeutig [86]. Das Feed-Forward-Netz verfügt über eine zeitliche Entfaltung der Stufe τ.

Die Länge der notwendigen zeitlichen Entfaltung ist problemabhängig und wird entweder durch Vorwissen ermittelt oder durch Berücksichtigung der sogenannten maximalen intertemporalen Konnektivität. Da die Aufgabe des Schätzens der Werte y_t durch das RNN sich bis in die Gegenwart t verbessert, sich also der Fehler reduziert, wird man an dem Zeitpunkt, ab dem sich in der Vergangenheit der Fehler zwischen zwei Zeitschritten nicht mehr unterscheidet, die zeitliche Entfaltung abschließen [125].

Entscheidend ist das Konzept der sogenannten Shared-Weights und ihre Anwendung durch das Shared-Weights-Back-Propagation (oder Back-Propagation-Through-Time) [86]. Dabei muss, im Gegensatz zu gewöhnlichen Feed-Forward-Netzen, jeder zeitinvariante funktionale Zusammenhang nach den Gln. 4.75 und 4.76 über die gleiche Aktivierungsfunktion f und die gleichen Konnektoren A, B und C erfolgen.

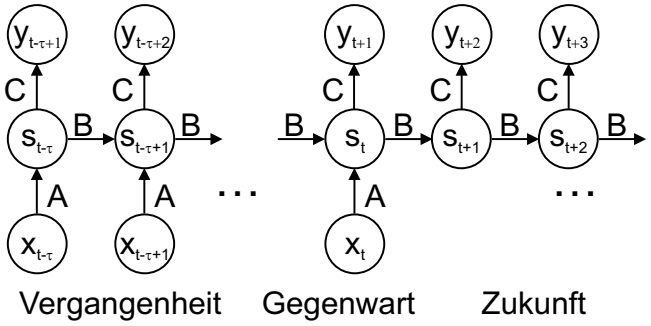


Abbildung 4.1: Rekurrentes neuronales Netz. Man unterscheidet zwischen internen Zuständen $\mathbf{x}_t$, externen Einflüssen s_t und Ausgabegrößen y_{t+1}, die i.d.R. s_{t+1} approximieren sollen. Im Overshooting-Teil (Zukunft) arbeitet das Netz autonom, das heißt, es werden keine externen Einflüsse mehr berücksichtigt.

4.4.2.3 Overshooting und dynamische Konsistenz

Das oben beschriebene Modell ist theoretisch hinreichend zur Beschreibung offener dynamischer Systeme. Zur erhöhten Gewichtung der autonomen Entwicklung der Dynamik und als Regularisierungstechnik kann man die Dynamik in die Zukunft fortsetzen, jedoch ohne weitere externe Umwelteinflüsse als Eingaben zu berücksichtigen [125]. Die Aufgabe, die y_t in der Zukunft vorherzusagen wird dadurch jedoch schwieriger.

Neben der genaueren Beschreibung der Dynamik und der Regularisierung hat dieses als Overshooting bezeichnete Vorgehen den Vorteil, dass die Ergebnisse y_t der Zukunft für die zu lösende Aufgabe genutzt werden können. Daher wird insbesondere in Prognoseaufgaben die Rolle der rekurrenten Netze noch bedeutender. Die Länge des Overshootings orientiert

sich grundsätzlich an der maximalen intertemporalen Konnektivität, kann sich aber auch problem- und aufgabenangepasst davon unterscheiden.

Alternativ kann das Overshooting auch dynamisch konsistent realisiert werden [124]. Dies ist nur bei Prognoseaufgaben möglich, bei denen wenigstens ein Teil der Ausgaben y_{t+1} semantisch mit den Eingaben s_{t+1}, also der zum nachfolgenden Zeitpunkt, übereinstimmt. Dabei werden im Overshooting-Teil des RNN die Ausgaben y_{t+1} mit den Eingaben s_{t+1} verbunden, so dass die eigene Prognose weiterpropagiert wird. Datenkonsistentes Overshooting hat offensichtlich den entscheidenden Vorteil, dass das Konzept der Shared-Weights, das auf den Overshooting-Teil übertragen wird, konsistent ist. Die auf diese Weise entstehenden rekurrenten neuronalen Netze werden als Dynamical-Consistent-Neural-Networks bezeichnet.

4.5 Theoretische Betrachtungen

Schließlich sollen einige theoretische Ergebnisse skizziert werden, die i.w.S. mit der Problematik der Dateneffizienz im Reinforcement-Learning zusammenhängen.

4.5.1 Performance-Schranken für Policy-Iteration

Wir betrachten Fitted-Value-Iteration im Falle kontinuierlicher Zustandsräume so wie in Abschn. 4.1 beschrieben. Bertsekas und Tsitsiklis haben bereits in [13] die Schranke

$$\left\|V^* - V^{\pi^k}\right\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}\left\|V^k - V^{\pi^k}\right\|_\infty \tag{4.77}$$

für $k \to \infty$ bewiesen. Dabei ist V^* die wahre optimale Bewertungsfunktion, V^{π_k} die wahre Bewertungsfunktion der Policy π^k, die nach der k-ten Iteration angewendet würde und V^k die nach der k-ten Iteration geschätzte Bewertungsfunktion. Diese Schranke ermöglicht also die Abschätzung der Performance der ermittelten Policy π^k, die demnach also gerade dann performanter wird, wenn der Fehler der geschätzten und der wahren Bewertungsfunktion für π^k niedrig ist. Diese Schranke zeigt eindrucksvoll die Notwendigkeit, die Generalisierungsfähigkeit von RL-Verfahren zu optimieren, damit die rechte Seite von Gl. 4.77, die unmittelbar beeinflusst werden kann, möglichst klein wird.

In [66] ist diese Schranke substanziell verbessert worden, indem die Maximums-Norm durch die euklidische Norm ausgetauscht werden konnte. Munos hat zeigen können, dass für beliebige Verteilungen μ Verteilungen μ_k und μ'_k existieren, die er in [66] explizit angegeben hat, so dass

$$\left\|V^* - V^{\pi^k}\right\|_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\left\|V^k - V^{\pi^k}\right\|_{\mu_k} \tag{4.78}$$

$$\left\|V^* - V^{\pi^k}\right\|_\mu \leq \frac{2\gamma}{(1-\gamma)^2}\left\|V^k - T^{\pi^k}V^k\right\|_{\mu'_k} \tag{4.79}$$

für $k \to \infty$, wobei $\|\cdot\|_\mu$ die gewichtete euklidische Norm unter der Verteilung μ bezeichnet. Dadurch ist Gl. 4.78 eine direkte Verbesserung von Gl. 4.77, wohingegen sich Gl. 4.79 auf das Bellman-Residuum bezieht (siehe Abschn. 2.5). Man beachte, dass ohne weitere Bemühungen aus Gl. 4.77 durch die Äquivalenz der Normen bereits

$$\left\|V^* - V^{\pi^k}\right\| \leq \frac{2\sqrt{n}\gamma}{(1-\gamma)^2}\left\|V^k - V^{\pi^k}\right\| \tag{4.80}$$

folgt, mit n der Anzahl der Zustände und der Annahme uniformer Verteilungen. In der Praxis ist Gl. 4.80 jedoch kaum anwendbar, da n sehr groß werden kann, evtl. sogar unendlich groß.

Gordon hat in [36] bewiesen, dass

$$\left\|V^* - V^k\right\|_\infty \leq \frac{2\gamma}{1-\gamma}\min_{V\in H}\left(\left\|V^* - V\right\|_\infty\right) \tag{4.81}$$

für $k \to \infty$, jedoch unter der Annahme, dass der MDP und damit P und R vollständig bekannt sind. Die Performance wird durch die Darstellbarkeit der optimalen Bewertungsfunktion innerhalb von H beschränkt. Diese Aussage zeigt die Notwendigkeit der sorgfältigen Auswahl des Hypothesenraumes H.

4.5.2 Kapazitätsschranken

Theoretische Ergebnisse, die sich auf die Menge verfügbarer Daten und Eigenschaften des Hypothesenraumes beziehen, liegen bisher nur wenige vor. Kearns et al. haben in [51] Schranken für die Anzahl der notwendigen Beobachtungen im Kontext von POMDP vorgestellt. Für Aktionsräume bestehend aus zwei Aktionen sind demnach gerade

$$m \quad = \quad O\left(\left(\frac{V_{\max}}{\varepsilon}\right)^2 \left(H_\varepsilon \mathbf{VCD}(H_\pi) + \log\left(\frac{1}{\delta}\right)\right)\right) \tag{4.82}$$

sogenannte Trajektorienbäume (Trajectory-Trees) der Tiefe H_ε ab einem Startzustand s_0 notwendig, um

$$\left|V^\pi(s_0) - \tilde{V}^\pi(s_0)\right| \quad \leq \quad \varepsilon \tag{4.83}$$

mit Wahrscheinlichkeit $P = 1-\delta$ zu garantieren. Dabei ist V^π die wahre und $\tilde{V}^\pi$ die geschätzte Bewertungsfunktion der Policy π, $H_\varepsilon = \log_\gamma\left(\frac{\varepsilon(1-\gamma)}{2R_{\max}}\right)$ die sogenannte ε-Horizont-Zeit, nach der der Reward höchstens noch um ε in den diskontierten Wert der Bewertungsfunktion eingeht und jede Policy $\pi \in H_\pi$ eine Abbildung von einer Menge an vergangenen Beobachtungen in den Aktionsraum. Ein Trajektorienbaum besteht dabei aus 2^{H_ε} Knoten, bei dem jede Kante eine der möglichen Aktionen zu den Zeitpunkten t, die der jeweiligen Tiefe der ausgehenden Knoten entspricht, darstellt. Jede Kante ist dabei mit dem vereinnahmten Reward versehen. Bezieht man das Resultat also auf die Anzahl der Einzelbeobachtungen, erhält man die erforderliche Mindestanzahl

$$k \quad = \quad O\left(2^{H_\varepsilon} m\right). \tag{4.84}$$

Andere Kapazitätsschranken sind von Peshkin et al. bewiesen worden [72]. Antos et al. haben erstmals eine umfassende Kapazitätsanalyse des Bellman-Residuums durchgeführt [3, 4]. Die Analyse beruht auf der Einführung des Konzeptes der sogenannten VC-Crossing-Dimension und der Modifikation des Bellman-Residuums, die eine Abschätzung des wahren Bellman-Residuums ermöglicht. Motiviert ist die Betrachtung des Bellman-Residuums gegenüber der TD-Lösung durch die in [66] bewiesenen Performance-Schranken, die sich für beide Fehlermaße qualitativ ähneln. Da jedoch der TD-Fehler stets mindestens so groß ist wie das minimale Bellman-Residuum, besteht die Aussicht, für das Bellman-Residuum bessere Performance-Schranken zu gewinnen. In [3, 4] konnte schließlich eine Schranke für den Ausdruck $\|Q^* - Q^{\pi_k}\|$, also den Performance-Unterschied zwischen der wahren optimalen und der wahren Q-Funktion der Policy π_k, die nach k Schritten ermittelt wurde, bewiesen werden, die linear im Bellman-Residuum und polynomiell in der durch die VC-Crossing-Dimension ausgedrückten Kapazität ist.

Teil II

Regularisierungsperspektive – Generalisierung mit Rewards-Regression

Kapitel 5

Reinforcement-Learning als Regression

In diesem Teil stellen wir den Rewards-Regression-Ansatz als Verallgemeinerung und Erweiterung der Fitted-Value-Iteration vor. Wir untersuchen die Rewards-Regression in Kombination mit Kernel-Machines und neuronalen Netzen als Funktionsapproximatoren. In beiden Fällen wird ein besonderer Schwerpunkt auf die Problematik der Regularisierung gelegt.

5.1 Übersicht

Zunächst beschäftigen wir uns in diesem Zusammenhang mit grundsätzlichen Fragestellungen, die mit der Steigerung der Informationseffizienz und der Verbesserung der Erwartungstreue der Zielfunktionen zusammenhängen. Insbesondere untersuchen wir die theoretischen Möglichkeiten, das Bellman-Residuum im Batch-Learning-Fall als Optimalitätskriterium heranzuziehen [94] und nehmen eine Einordnung der später vorzustellenden Verfahren vor. Im Anschluss daran erläutern wir die Kernel-Rewards-Regression und die Neural-Rewards-Regression.

5.1.1 Motivation und Prinzip der Rewards-Regression

Die Grundidee der Rewards-Regression besteht in der Auffassung von Reinforcement-Learning-Problemen als Regressionsprobleme. Jedoch werden, im Gegensatz zur Fitted-Value-Iteration, bei der iterativ für Policy-Evaluation und Policy-Iteration die Regressionsprobleme

$$(T^\pi Q)(s,a) = \mathbf{E}_{s'}(R(s,a,s') + \gamma Q(s', \pi(s'))) \tag{5.1}$$

$$(TQ)(s,a) = \mathbf{E}_{s'}\Big(R(s,a,s') + \gamma \max_{a'} Q(s',a')\Big) \tag{5.2}$$

gelöst werden, diese so umformuliert, das man zu einer direkten Lösung gelangt und sie sich als

$$\mathbf{E}_{s'}(R(s,a,s')) = (T^\pi Q)(s,a) - \gamma \mathbf{E}_{s'}(Q(s', \pi(s'))) \tag{5.3}$$

$$\mathbf{E}_{s'}(R(s,a,s')) = (TQ)(s,a) - \gamma \mathbf{E}_{s'} \max_{a'}(Q(s',a')) \tag{5.4}$$

schreiben lassen. Die Aufgabe besteht also darin, die rechte Seite der Gln. 5.3 bzw. 5.4 als Reward-Funktion auf beobachtete Rewards abzubilden. Dabei muss der Hypothesenraum H_R der Funktionen $R : S \times A \times S \to \mathbb{R}$ mit $R \in H_R$ so gewählt werden, dass die Bellman-Gleichung bzw. die Bellman-Optimalitäts-Gleichung eingehalten wird und die Q-Funktion daraus hervorgeht. Das RL-Problem wird so in ein reines Supervised-Learning-Problem überführt.

Die Motivation für die Rewards-Regression besteht in den folgenden Aspekten. Wie bei der Fitted-Value-Iteration kommt Regression zum Einsatz, wodurch die Anwendung mächtiger Instrumentarien zur Regularisierung und Generalisierung ermöglicht wird. Darüber hinaus wird das RL-Problem in ein reines Supervised-Learning-Problem umgewandelt. Daher gehen wir von einer besseren Beherrschbarkeit des Problems und geringerer Auswirkungen des Policy-Degradation-Phänomens aus [29]. Zudem eröffnet diese Formulierung des Problems eine Reihe von Erweiterungsmöglichkeiten, die sich bei der Fitted-Value-Iteration nur bedingt umsetzen lassen.

Da im Falle einer gültigen Lösung $Q = TQ$ gilt und wir tatsächlich die Werte der beobachteten Folgezustände s' und Rewards r als erwartungstreue Schätzer der Ergebnisse der Transitionen und der Reward-Funktion R heranziehen können, gilt für die Gln. 5.3 und 5.4 im Erwartungswert

$$R(s,a,s') = Q(s,a) - \gamma Q(s', \pi(s')) \tag{5.5}$$
$$R(s,a,s') = Q(s,a) - \gamma \max_{a'} (Q(s',a')). \tag{5.6}$$

Diese Transformation gilt exakt nur in den Fällen, für die die wahre Reward-Funktion in H_R enthalten ist. Anderenfalls hängt es von der verwendeten Zielfunktion ab, inwieweit die Erwartungstreue gewährleistet ist. Wir werden bei der KRR (Abschn. 6.3) und der NRR (Kap. 7 und Abschn. 7.3) auf diese Problematik jeweils im Detail eingehen.

5.1.2 Kernel-Rewards-Regression

Die Kernel-Rewards-Regression (KRR) [98] (Kap. 6) ist eng an die Least-Squares-Policy-Iteration [57] angelehnt (siehe Abschn. 4.1), verwendet jedoch eine stochastische Policy-Iteration-Technik, mit der wir Konvergenz garantieren und ist vor allem eine kernelisierte Ausprägung (siehe Abschn. 3.4). Wir stellen die Optimalitätskriterien Fixpunkt der Bellman-Iteration und Minimum des Bellman-Residuums gegenüber (siehe Abschn. 2.5) und untersuchen ihre speziellen Auswirkungen beim Approximieren mit Kernels.

Im Anschluss erweitern wir die KRR um Support-Vector-Machines und stellen einen Ansatz vor, der das Prinzip der SVM nutzt und ohne Policy-Iteration mittels quadratischer Programmierung eine Lösung bestimmt [99]. Wir diskutieren ferner, inwieweit kontinuierliche Aktionsräume verwendet werden können, wie die Problemklasse auf POMDP und MDP höherer Ordnung erweitert werden kann [97] und inwiefern die KRR sowohl als modellfreies als auch als modellbasiertes Verfahren aufgefasst werden kann.

5.1.3 Neural-Rewards-Regression

Die Neural-Rewards-Regression (NRR) [101] (Kap. 7) kann dagegen als eine Verallgemeinerung der Neural-Fitted-Q-Iteration [83] aufgefasst werden. Jedoch besteht auch hier die Möglichkeit, unterschiedliche Iterationstechniken einzusetzen, darüber hinaus ist die NRR eine Verallgemeinerung hinsichtlich der beiden oben genannten Optimalitätskriterien. Im Anschluss daran untersuchen wir eine Möglichkeit, die Erwartungstreue der zu Grunde liegenden Zielfunktionen zu verbessern, indem wir den Ansatz aus [3, 4] in neuronalen Netzen implementieren und die Erweiterungsmöglichkeit für kontinuierliche Aktionsräume und POMDP sowie MDP höherer Ordnung. Dies resultiert in die Verfahren Policy-Gradient-Neural-Rewards-Regression (PGNRR) [100] und Recurrent-Neural-Rewards-Regression (RNRR) [101], die auch in Kombination verwendet werden können. Wir untersuchen für die RNRR das theoretische Potenzial mit POMDP umzugehen [101] und für die PGNRR die weitere Steigerung der Informationseffizienz durch eine verschränkte Regularisierung von Q-Funktion und Policy [100].

5.1.4 Ergebnisse

Zur Demonstration der Informationseffizienz der Verfahren folgen in Kap. 8 schließlich Benchmark-Ergebnisse, das auch Vergleiche mit anderen Verfahren enthält sowie der praktische Nachweis der Verbesserung der Generalisierungsleistung durch verschränkte Regularisierung beim PGNRR. Außerdem werden diese Techniken auf Simulationen zur Regelung von Gasturbinen angewendet mit dem Ziel, ihre industrielle Einsatzfähigkeit nachzuweisen.

5.2 Garantierte Performance bei der Rewards-Regression

Eine Reihe von Performance-Schranken für Reinforcement-Learning, insbesondere für Policy-Iteration liegen bereits vor (siehe Abschn. 4.5). Wir wollen ein einfaches Resultat zeigen, dass sich an der Qualität der Regression an den Rewards orientiert.

Satz 4. *[98] Sei S ein kompakter Zustandsraum und A ein endlicher Aktionsraum. Seien ferner $\bar{R}$, $\bar{Q}$ die wahren erwarteten Rewards und die wahre Q-Funktion für eine Policy π sowie R, Q deren Schätzer, die den Zusammenhang*

$$R(s,a,s') \quad = \quad Q(s,a) - \gamma Q(s', \pi(s')) \tag{5.7}$$

für alle $s, s' \in S$ und $a \in A$ mit $0 < \gamma < 1$ als Diskontierungsfaktor erfüllen. Ist nun für alle $s, s' \in S$ und $a \in A$ der Fehler

$$\left|\bar{R}(s,a,s') - R(s,a,s')\right| \quad < \quad \varepsilon \tag{5.8}$$

beschränkt, gilt der Zusammenhang

$$\left|\bar{Q}(s,a) - Q(s,a)\right| \quad \leq \quad \frac{\varepsilon}{1-\gamma}. \tag{5.9}$$

Beweis: Angenommen, es gelte das Gegenteil, es existiert mindestens ein (s^*, a^*), so dass

$$\frac{\varepsilon}{1-\gamma} \quad < \quad \left|(\bar{Q} - Q)(s^*, a^*)\right|. \tag{5.10}$$

Wegen der Gültigkeit der Bellman-Gleichung für $\bar{Q}$ und $\bar{R}$ sowie Gl. 5.7 für Q und R, existiert wenigstens ein (s,a), das den Fehler der Q-Funktion maximiert und

$$\frac{\varepsilon}{1-\gamma} + c \quad = \quad \left|\left(\bar{Q} - Q\right)(s,a)\right| \tag{5.11}$$

$$= \quad \left|\mathbf{E}_{\hat{s}}\left(\left(\bar{R}\left(s,a,\hat{s}\right) + \gamma \bar{V}\left(\hat{s}\right)\right) - \left(R\left(s,a,\hat{s}\right) + \gamma V\left(\hat{s}\right)\right)\right)\right| \tag{5.12}$$

mit $c > 0$. Dann gibt es mindestens ein s', so dass

$$\frac{\varepsilon}{1-\gamma} + c \quad \leq \quad \left|\left(\bar{R} - R\right)(s,a,s') + \gamma\left(\bar{V} - V\right)(s')\right|. \tag{5.13}$$

Darüber hinaus folgt aus $\left|\left(\bar{V} - V\right)(s')\right| = \beta$, dass $\exists a' \in A : \left|\left(\bar{Q} - Q\right)(s', a')\right| \geq \beta$. Also erhalten wir direkt

$$\begin{aligned}
\frac{\varepsilon}{1-\gamma} + c &\leq \left|\left(\bar{R} - R\right)(s,a,s') + \gamma\left(\bar{Q} - Q\right)(s',a')\right| && (5.14)\\
&\leq \left|\left(\bar{R} - R\right)(s,a,s')\right| + \gamma\left|\left(\bar{Q} - Q\right)(s',a')\right| && (5.15)\\
&< \varepsilon + \gamma\left(\frac{\varepsilon}{1-\gamma} + c\right) && (5.16)\\
&= \frac{\varepsilon}{1-\gamma} + \gamma c. && (5.17)
\end{aligned}$$

Dies ist ein Widerspruch und beweist das Theorem.

□

Satz 4 ist in der Praxis natürlich schwer anwendbar, da aus der Interpolationsgüte auf den Beobachtungen nur indirekt Aussagen über die Approximationsgüte, also die Generalisierungsfähigkeit gemacht werden können. Da aber konkrete Schranken, insbesondere auch Kapazitätsschranken, für Regressionsverfahren vorliegen, kann das Theorem zusammen mit ihnen angewendet werden. Es zeigt sich quantitativ die Notwendigkeit einer guten Approximation an die Reward-Funktion, da ihre Approximationsgüte linear mit der der Q-Funktion zusammenhängt.

5.3 Kategorisierung der Verfahren

Die Rewards-Regression ist eine Verallgemeinerung der Fitted-Value-Iteration und ordnet sich mit ihren speziellen Ausführungen in die Klasse der Reinforcement-Learning-Verfahren ein. Dabei betrachten wir nur solche Methoden, die RL als Batch-Learning realisieren und unterscheiden hinsichtlich folgender Kriterien.

- Iterationstechnik (Iteration über die Bewertungsfunktion, direkte Policy-Suche)
- Klasse des Approximators und statistisches Paradigma
- Angestrebte Optimalität
- Beherrschbarkeit von MDP höherer Ordnung

Wir verzichten auf die Betrachtung der Dimensionen On-Policy und Off-Policy sowie modellfrei und modellbasiert. Die Einordnung der Rewards-Regression als modellfreies und modellbasiertes Verfahren wird bereits in Abschn. 6.1 diskutiert. Darüber hinaus müssen wir wegen der Fokussierung auf Batch-Learning den Einsatz von Off-Policy-Verfahren fordern. Wie wir bereits in Abschn. 2.4.2 gesehen haben, stellt dies keine Einschränkung an Mächtigkeit oder Anwendbarkeit dar. Im Gegenteil, Off-Policy-Verfahren stellen weniger Anforderungen an die Exploration, was eine notwendige Voraussetzung für die Beherrschbarkeit der in der vorliegenden Arbeit betrachteten breiteren Problemklasse ist.

Die Tabn. 5.1 und 5.2 geben einen groben Überblick über die Vielfalt der Aspekte, die alleine in unserer Betrachtungsweise eine Rolle spielen und ermöglichen, den Rewards-Regression-Ansatz einzuordnen. Die Methoden BPG, KBRL, RCNN und PGRL sowie KRR, NRR, RNRR, PGNRR, RPGNRR stehen dabei für Bayesian-Policy-Gradient [33], kernel-basiertes Reinforcement-Learning, Recurrent-Control-Neural-Network [93] und Policy-Gradient-RL, Kernel-Rewards-Regression, Neural-Rewards-Regression, Recurrent-NRR, Policy-Gradient-NRR und Recurrent-Policy-Gradient-NRR.

5.4 Batch-Reinforcement-Learning und das Bellman-Residuum

Wir betrachten erneut die Fragestellung des Minimierens des Bellman-Residuums als zum Fixpunkt der Bellman-Iteration alternativen Optimierungskriteriums und greifen damit die Ausführungen aus Abschn. 4.2 auf. Wir motivieren bzw. zeigen drei Beobachtungen.

Tabelle 5.1: Einordnung verschiedener Reinforcement-Learning-Verfahren nach Funktionsapproximatoren und Iterationstechniken. Mit Funktionsapproximatoren ist hier die Klasse von Funktionen zusammen mit dem Approximationsverfahren gemeint, die Iterationstechnik bezieht sich auf die konkrete Implementierung der Policy-Iteration.

	Iterationstechnik				
Funktions-approximator	Direkt	Vollständige Inversion	Fixpunkt-Iteration	Iterative Fix-punktsuche	Policy-Suche
Diskret	Lineare Program-mierung	DP mit Gauß-Seidel-Iteration	DP	entf.	entf.
Lineare Architekturen	entf.	LSPI, vollstän-dige Matrix-inversion	LSPI, itera-tive Matrix-inversion	LSPI, iterativ	PGRL
Kernel-Machines	EKRR	Ridge-KRR, SVRR	KBRL	entf.	entf.
Gauß-Prozesse	entf.	GPTD	entf.	entf.	BPG
Reservoir-Computing	EKRR mit Reservoirs	KRR mit Reservoirs	KRR mit Reservoirs	entf.	entf.
Neuronale Netze	NRR	Policy Iteration mit NN	NFQ	entf.	RCNN

1. Das Auxiliared-Bellman-Residuum ist kein konsistenter Schätzer für das wahre Bellman-Residuum.

2. Es existiert kein erwartungstreuer Schätzer für das Bellman-Residuum im Batch-Learning-Fall.

3. Das Minimum des Auxiliared-Bellman-Residuums ist unter bestimmten Voraussetzungen ein Fixpunkt der Bellman-Iteration.

Wir betrachten in der hier verwendeten Notation die Darstellung von Fehlerfunktionen $L(s,a) = f(Q,s,a)$ auf einer Beobachtung oder $L = g(Q)$ (z.B. $L = \|f(Q)\|^2$) auf einer Verteilung von Beobachtungen. Der tatsächliche Schätzer für eine Fehlerfunktion, etwa das Bellman-Residuum, ist schließlich über eine endliche Menge von Beobachtungen definiert und ergibt sich zu $L = \sum_{i=1}^{l} L(s_i, a_i)$.

Wie wir bereits in Abschn. 4.2 gesehen haben, ist $Y(s,a,s') = R(s,a,s') + \gamma V(s')$ zwar ein erwartungstreuer Schätzer für $(TQ)(s,a)$, jedoch ist der sample-basierte Schätzer $L(s,a)$ kein erwartungstreuer Schätzer für das wahre Bellman-Residuum, das TQ enthält, da

$$\begin{aligned}
\mathbf{E}_{s'} L(s,a) &= \mathbf{E}_{s'}(Q(s,a) - Y(s,a,s'))^2 && (5.18)\\
&= \mathbf{E}_{s'}\big(Q(s,a)^2 - 2Q(s,a)Y(s,a,s') + Y(s,a,s')^2\big) && (5.19)\\
&= Q(s,a)^2 - 2Q(s,a)\mathbf{E}_{s'}Y(s,a,s') + \mathbf{E}_{s'}Y(s,a,s')^2 && (5.20)\\
&= Q(s,a)^2 - 2Q(s,a)(TQ)(s,a) + \big((TQ)(s,a)^2 + (T'Q)(s,a)^2\big) && (5.21)\\
&= (Q(s,a) - (TQ)(s,a))^2 + (T'Q)(s,a)^2. && (5.22)
\end{aligned}$$

Das Minimieren der Funktion L schließt das Minimieren der Varianz des Gesamtertrags ein, was als zusätzlicher Regularisierungsterm gewertet werden kann. Wir bezeichnen L daher als deterministisches Bellman-Residuum, da es nur für deterministische MDP einem erwartungstreuen Schätzer für das wahre Bellman-Residuum entspricht.

Tabelle 5.2: Einordnung verschiedener Reinforcement-Learning-Verfahren nach Optimalität und Problemklassen. Die Problemklasse bezieht sich auf die Klasse von MDP, die das jeweilige Verfahren beherrscht, während sich Optimalität auf das angestrebte Optimalitätskriterium bezieht.

Optimalität	**Problemklasse**			
	MDP		POMDP	
	Diskrete Aktionen	Kontinuierliche Aktionen	Diskrete Aktionen	Kontinuierliche Aktionen
TD-Lösung	DP, LSPI, KBRL, NFQ, NRR, GPTD	PGNRR	RNRR, RLSPI	RPGNRR
Residual-Lösung	NRR	PGNRR	RNRR	RPGNRR
Monte-Carlo-Lösung	PGRL	PGRL	RPGRL	RCNN
Large-Margin-TD-Lösung	KRR mit SVM	entf.	entf.	entf.
Large-Margin-Res.-Lösung	KRR mit SVM	entf.	entf.	entf.

5.4.1 Spezialfälle für die Schätzung des Bellman-Residuums

Unabhängig davon gibt es mindestens drei Spezialfälle, für die konsistente und erwartungstreue Schätzer für das Bellman-Residuum, selbst im stochastischen Fall, angegeben werden können. Im ersten Fall ist eine analytische Funktion $\mathrm{Var}(Q,s,a) = (T'Q)(s,a)^2$ a priori bekannt. Dann betrachtet man

$$\begin{aligned} L_V(s,a) &= L(s,a) - \mathrm{Var}(Q,s,a) \\ &= (Q(s,a) - R(s,a,s') - \gamma V(s'))^2 - \mathrm{Var}(Q,s,a). \end{aligned} \tag{5.23}$$

Deterministische MDP, mit $\forall s,a : \mathrm{Var}(Q,s,a) = 0$, sind ein besonderer Spezialfall, der hiermit abgedeckt wird.

Im zweiten Fall ist ein erwartungstreuer Schätzer für die Varianz jeder Beobachtung gegeben. Dies ist ein angemessener Vorschlag, falls ein Modell des MDP bekannt ist oder jedes Zustands-Aktions-Paar bewusst mindestens zweimal beobachtet werden kann [107, 57, 6]. Dies haben wir bereits in Abschn. 4.2 ausgeführt.

Im dritten Fall ist der Funktionsraum H_Q so gewählt, dass $\forall s_i, s_i' \in S, a_i \in A, r_i \in \mathbb{R} : \exists Q \in H : \forall i : L(s_i,a_i) = 0$. Dann ist L tatsächlich ein erwartungstreuer Schätzer für das wahre Bellman-Residuum innerhalb von H_Q, mindestens für $Q = Q^*$. Es genügt, das Gleichungssystem $\forall i : Q(s_i,a_i) = Y(s_i,a_i,s_i')$, also für alle Beobachtungen, zu lösen. Darüber hinaus ist Q^* gleichzeitig die Fixpunkt-Lösung der Bellman-Iteration.

5.4.2 Der allgemeine Fall für die Schätzung des Bellman-Residuums

Für den allgemeinen Fall kommen wir auf den Vorschlag in [3, 4] zurück (siehe Abschn. 4.2), der im Lösen des modifizierten Bellman-Residuums

$$L_{\mathrm{aux}}(s,a) = L(s,a) - \hat{L}(s,a) \tag{5.24}$$

$$\begin{aligned} &= (Q(s,a) - \gamma V(s') - R(s,a,s'))^2 \\ &\quad - (h(s,a) - \gamma V(s') - R(s,a,s'))^2 . \end{aligned} \tag{5.25}$$

besteht. In diesem Zusammenhang definieren wir

$$L'(f,g) = (f(s,a) - \gamma g_V(s') - R(s,a,s'))^2 \tag{5.26}$$

mit $g_V(s) = \max_a g(s,a)$. Wie oben bereits beschrieben, wird h aus H_h gewählt. Antos et al. schlagen die Wahl $H_h = H_Q$ vor und argumentieren, dass, wenn der Fehler bzgl. h für TQ durch ein ε beschränkt werden kann, der Ausdruck $\mathbf{E}L_{\text{aux}} + \varepsilon$ das wahre Bellman-Residuum nach oben beschränkt. Wir erhalten dann folgende Schlussfolgerung.

Beobachtung 1. *[94] Seien L_{aux} und L' wie oben definiert mit $h = \arg\min_{h' \in H_h} L'(h', Q)$, $Q \in H_Q$, H_Q und H_h beliebig, aber fest gewählt, so dass die optimale Q-Funktion $Q^* = Q^*((s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l))$ und $h = h(Q, (s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l))$ eindeutig definiert sind. Dann kann L_{aux} das wahre Bellman-Residuum $\|Q - TQ\|^2$ im Allgemeinen nicht konsistent schätzen.*

Begründung: Angenommen, es gelte das Gegenteil und L_{aux} ist ein konsistenter Schätzer für $\|Q - TQ\|^2$. Wir definieren L und $\hat{L}$ wie oben und demnach $L_{\text{aux}} = L - \hat{L}$. Offensichtlich ist L ein konsistenter Schätzer für $\|Q - TQ\|^2 + \|T'Q\|^2$. Also ist $\hat{L}$ ein konsistenter Schätzer für $\|T'Q\|^2$. Dies führt jedoch direkt zu einem Widerspruch, da H_h ein eingeschränkter Funktionsraum ist. Für jedes beliebige H_h kann ein MDP M angegeben werden, so dass $TQ \notin H_h$. Mit $l \to \infty$ erhalten wir daher $\hat{L} \to \hat{L}_\infty > \|T'Q\|^2$.

□

Bemerkung: Beobachtung 1 schließt Konsistenz nicht zwangsläufig aus, sofern H_h nicht fest gewählt ist, sondern von l abhängt. Mit $l \to \infty$ können die Funktionsräume H_Q und H_h gegebenenfalls entsprechend erweitert werden, so dass $H_h \to H^\infty \ni TQ$.

Bemerkung: Ein konsistenter Schätzer für das wahre Bellman-Residuum existiert natürlich, etwa

$$L_C(s,a) \quad = \quad (Q(s,a) - Y(s,a,s'))\,(Q(s_u,a_u) - Y(s_u,a_u,s'_u))\,, \tag{5.27}$$

mit s_u und a_u der jeweils am dichtesten an s und a liegenden Beobachtungen, etwa in der euklidischen Norm. Mit $l \to \infty$ erhalten wir $(s_u, a_u) \to (s, a)$ per Konstruktion und schließlich $(TQ)(s_u, a_u) \to (TQ)(s, a)$. Jedoch hat der Schätzer, abhängig von $T'Q$ und der Anzahl der Beobachtungen einen großen Bias.

Eine stärkere Forderung als Konsistenz ist also Erwartungstreue. Wir zeigen, dass im Allgemeinen kein erwartungstreuer Schätzer für das Bellman-Residuum im Batch-Learning-Fall existiert.

Beobachtung 2. *[94] Sei $H_L \subset (H_Q \mapsto \mathbb{R})$ eine beliebige Funktionsklasse von Fehlerfunktionen, die von den beobachteten Zuständen, Aktionen, Rewards und Folgezuständen abhängen. Dann kann im Allgemeinen keine Funktion $L_B \in H_L$ ein erwartungstreuer Schätzer für das wahre Bellman-Residuum $\|Q - TQ\|^2$ sein.*

Begründung: Es ist bereits bekannt, dass L ein erwartungstreuer Schätzer für $\|Q - TQ\|^2 + \|T'Q\|^2$ ist, also $\mathbf{E}L = \|Q - TQ\|^2 + \|T'Q\|^2$. Nun betrachten wir $\hat{L} : H_Q \mapsto \mathbb{R}$ als einen Schätzer für $\|T'Q\|^2$. Dieser kann nur von $((s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l))$ abhängen. Daher kann $\hat{L}$ höchstens ein erwartungstreuer Schätzer für eine Funktion aus einem eingeschränkten Funktionsraum H_{Var} sein. Also kann ein MDP angegeben werden, für den $\|T'Q\|^2 \notin H_{\text{Var}}$. Nun nehmen wir das Gegenteil an, es existiert ein $L_B \in H_L$, so dass $\mathbf{E}L_B = \|Q - TQ\|^2$. Wegen der Linearität des Erwartungswertoperators erhalten wir $\mathbf{E}\hat{L} = \|T'Q\|^2 = (\|Q - TQ\|^2 + \|T'Q\|^2) - \|Q - TQ\|^2 = \mathbf{E}L - \mathbf{E}L_B$. Der Widerspruch folgt unmittelbar, da ein erwartungstreuer Schätzer für $\|T'Q\|^2$ existieren würde.

□

Wählt man nun tatsächlich einen angemessenen Funktionsraum, um den Bellman-Operator hinreichend präzise zu approximieren, dann bietet L_{aux} eine gute Approximation an das wahre Bellman-Residuum. Sie ist jedoch strukturell gebiased hinsichtlich ihrer Minimierungseigenschaften.

Satz 5. *[94] Seien L_{aux} und L' wie oben gegeben und $h = \arg\min_{h' \in H} L'(h', Q)$. Sei ferner H beliebig, aber fest gewählt, so dass $Q^* = Q^*((s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l))$, die optimale Q-Funktion und $h = h(Q, (s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l))$ eindeutig bestimmt sind. Dann ist das Minimum Q^* von L_{aux} gleichzeitig ein Fixpunkt der Bellman-Iteration, also $Q^* = \arg\min_{Q \in H} L'(Q, Q^*)$, wenn er existiert.*

Beweis: Da $Q, h \in H$, gilt $\hat{L}(Q) \leq L(Q)$. Andererseits würde die Wahl von $h = Q$ zu $\hat{L}(Q) = L(Q)$ führen und h wäre nicht optimal gewesen. Ferner kann $\hat{L}(Q) = L(Q)$ nur in dem Fall erreicht werden, falls $Q = h$, da h gerade $\hat{L}$ minimiert. Aber dann ist Q ein Fixpunkt der Bellman-Iteration wegen $Q = h = \arg\min_{h' \in H} L'(h', Q)$. Eine solche Lösung existiert, falls ein Fixpunkt existiert.

□

Dieses Ergebnis führt zu einer wichtigen nicht-trivialen Schlussfolgerung. Für den Spezialfall $H_Q = H_h$ entspricht das Auxiliared-Bellman-Residuum einer geschlossen darstellbaren Fehlerfunktion, deren Minimum zum Fixpunkt der Bellman-Iteration führt. Dass eine solche existiert, ist nicht offensichtlich, da für die Fixpunkt-Lösung die Richtung des Gradienten bzgl. des Bellman-Residuums (siehe Abschn. 2.5.2) so modifiziert wird, dass der Gradient nicht mehr zum Fehlerterm korrespondiert.

Kapitel 6

Kernel-Rewards-Regression

Bei der Kernel-Rewards-Regression wird das Prinzip der Rewards-Regression für Hypothesenräume angewendet, die auf Kernels beruhen. Dabei wird die Q-Funktion als

$$Q(s,a) = \sum_{i=1}^{l} \alpha_i K((s,a),(s_i,a_i)) \tag{6.1}$$

dargestellt, wobei die (s_i, a_i) Stützstellen sind, die i.d.R. mit den gemachten Beobachtungen identifiziert werden. Wir gehen zunächst davon aus, dass der Aktionsraum diskret ist und wählen K so, dass ein Bias daraus extrahiert werden kann und, der Einfachheit halber, die Q-Funktionen für jede Aktion separat definiert sind. Dann ergibt sich mit

$$\begin{aligned} K((s,a),(s_i,a_i)) &= \delta_{a,a_i} K(s,s_i) && (6.2)\\ &= \delta_{a,a_i}\left(\hat{K}(s,s_i)-1\right) && (6.3)\\ b_a &= \sum_{i=1}^{l} \delta_{a,a_i}\alpha_i && (6.4) \end{aligned}$$

die Q-Funktion als

$$Q(s,a) = \sum_{i=1}^{l} \alpha_i \delta_{a,a_i} \hat{K}(s,s_i) - b_a. \tag{6.5}$$

Das Konzept der Kernelisierung erfolgt dabei durch die Interpretation des Kernels als Skalarprodukt in einem geeigneten Feature-Raum, so dass $K(s,\hat{s}) = \Phi(s)^T\Phi(\hat{s})$. Denn dann ergibt sich

$$Q(s,a) = \sum_{i=1}^{l} \alpha_i \delta_{a,a_i} \Phi(s)^T \Phi(s_i) \tag{6.6}$$

und damit die für jede Aktion a implizit gegebenen Gewichtsvektoren $\mathbf{w}_a$ zu

$$\mathbf{w}_a = \sum_{i=1}^{l} \alpha_i \delta_{a,a_i} \Phi(s_i). \tag{6.7}$$

In Matrixschreibweise fassen wir $\Phi, \Phi' \in \mathbb{R}^{l\times d}$ als die Feature-Matrizen der Ausgangs- und der Folgezustände auf und es ergibt sich für den Policy-Evaluation-Fall schließlich

$$\begin{aligned} V &= \Phi\mathbf{w} && (6.8)\\ &= R + \gamma V' && (6.9)\\ &= R + \gamma\Phi'\mathbf{w}, && (6.10) \end{aligned}$$

das zur Least-Squares-Lösung durch Lösen des Gleichungssystems

$$\Phi^T \Phi \mathbf{w} = \Phi^T (R + \gamma \Phi' \mathbf{w}) \tag{6.11}$$

führt. Durch Multiplikation mit Φ ergibt sich nach Substitution von $K = \Phi\Phi^T$ und $K\alpha = \Phi\mathbf{w}$ die Kernelisierung der Bellman-Gleichung zu

$$\Phi\Phi^T \Phi \mathbf{w} = \Phi\Phi^T (R + \gamma \Phi' \mathbf{w}) \tag{6.12}$$

$$\Leftrightarrow KK\alpha = K(R + \gamma K'\alpha). \tag{6.13}$$

Dies resultiert aus der Tatsache, dass die Kernel-Matrix als Außenprodukt der Feature-Matrix geschrieben werden kann. Ist der Kernel regulär, also ist K invertierbar, folgt unmittelbar

$$K\alpha = R + \gamma K'\alpha \tag{6.14}$$

$$\Leftrightarrow \alpha = (K - \gamma K')^{-1} R. \tag{6.15}$$

6.1 Kernel-Rewards-Regression als modellfreier und modellbasierter Ansatz

Diese Darstellung lässt sich auf zweierlei Art und Weise motivieren. Auf der einen Seite entspricht sie einer kernelisierten Formulierung der Q-Funktion entsprechend der LSPI [57] (siehe Abschn. 4.1) mit Bias b_a. Auf der anderen Seite ermöglicht sie auch eine dem kernel-basierten Reinforcement-Learning [69] (siehe Abschn. 4.1) vergleichbare Interpretation. Wenn K als abstandsbasierter Kernel gewählt wird, entspricht er einem Ähnlichkeitsmaß. Dies führt zu der Interpretation

$$\int_{s' \in S} P(s'|s,a) V(s') ds' \approx \sum_{i=1}^{l} K((s,a),(s_i,a_i)) V(s'_i) \tag{6.16}$$

$$= \sum_{i=1}^{l} \beta_i K((s,a),(s_i,a_i)) \tag{6.17}$$

und damit zu der Identifikation $\alpha_i = R(s_i,a_i,s'_i) + \gamma\beta_i = R(s_i,a_i,s'_i) + \gamma V(s'_i)$. Der Kernel liefert, zusammen mit den Beobachtungen, eine Approximation der Übergangswahrscheinlichkeitsdichtefunktion. Damit qualifiziert sich KRR als modellbasiertes Reinforcement-Learning, da der Kernel ein implizites Modell der Dynamik realisiert.

Dieses Modell betrachtet ausschließlich Übergänge in die Zustände $(s'_1, \ldots, s'_l)$ und P wird daher, von der Normierung von K abgesehen, mittels

$$P(s'_i|s,a) = K((s,a),(s_i,a_i)) \tag{6.18}$$

approximiert. Wird K ohne Vorwissen als abstandsbasierter Kernel, etwas als Gauß-Kernel, gewählt, dann wird die Wahrscheinlichkeit für den Übergang von s über a nach s'_i gerade dann als hoch eingeschätzt, wenn das Zustands-Aktions-Paar (s,a) gerade (s_i,a_i) ähnlich ist, da der Folgezustand s' auch mit hoher Wahrscheinlichkeit s'_i ähnlich ist. Wenn also detailliertes Wissen über den MDP vorliegt, kann der Kernel entsprechend gezielt gestaltet werden.

Die Wahl des Kernels kann daher auf der Grundlage zweier verschiedener Aspekte geschehen. Liegt Vorwissen über die Gestalt der Bewertungsfunktion oder der Q-Funktion vor, sollte der Kernel so gewählt werden, dass er die Darstellung einer linearen Q-Funktion im Feature-Raum ermöglicht, bei Vorwissen über die Gestalt der Übergangswahrscheinlichkeiten kann der Kernel entsprechend modelliert werden, bei abstandsbasierten Kernels etwa durch Festlegung der Varianz, die den Grad der Glättung der Dichtefunktion regelt.

Bei der Approximation der Q-Funktion mit neuronalen Netzen, wie es bei der Neural-Fitted-Q-Iteration und der Neural-Rewards-Regression der Fall ist (siehe Abschn. 7.5), lässt sich ähnlich argumentieren, auch wenn sich durch die Nichtlinearität des Approximators die Transitionswahrscheinlichkeiten nicht explizit darstellen lassen. Das Modell des MDP ergibt sich aber auch hier direkt aus den Daten.

Zur Frage, ob modellfreies oder modellbasiertes Reinforcement-Learning vorzuziehen ist, sind dogmatische Argumentationslinien also abzulehnen. Die Dualität der Kernel-Rewards-Regression veranlasst uns, die Position zu vertreten, dass zumindest im Batch-Learning-Fall eine solche Unterscheidung prinzipiell nicht möglich ist, bzw. sich das Modell unmittelbar aus den Daten ergibt und durch diese repräsentiert wird. TD-Learning etwa kann kein modellbasiertes Verfahren sein, da es sich die beobachteten Daten nicht aufhebt.

Natürlich liegt es nahe zu argumentieren, dass das Vorhandensein eines approximativen Modells der Umgebung von Vorteil ist, da es genutzt werden kann, um weitere Trajektorien zu generieren oder als Grundlage für ein an die dynamische Programmierung angelehnten Verfahrens dienen kann. Jedoch wird die direkte Bestimmung einer Bewertungsfunktion oder einer Policy insoweit von Vorteil sein, als die verfügbaren Daten anderenfalls bereits zur Modellbestimmung genutzt worden wären, so dass Bewertungsfunktion und Policy bereits von einem nicht vorurteilsfreien Modell ausgehen müssen. Daher kommen zwei verschiedene Bias-Formen ins Spiel und es scheint eher sinnvoll zu sein, auf erstere zu verzichten, denn das endgültige Ziel besteht nicht darin, ein möglichst gutes Modell zu konstruieren, sondern eine möglichst performante Policy zu generieren.

Im Rahmen einer ausgeglichenen Regularisierung kann es jedoch sinnvoll sein, den Bias auf verschiedene Approximationsstufen aufzuteilen, wie wir in Abschn. 7.4.1 an der verschränkten Regularisierung von Bewertungsfunktion und Policy noch experimentell zeigen werden. Dies ist insbesondere dann interessant, wenn Vorwissen über die beteiligten Komponenten Modell, Bewertungsfunktion und Policy zur Verfügung stehen.

6.2 Ridge-Kernel-Rewards-Regression

Das Problem der Regression an die Rewards für die KRR besteht schließlich in der Überführung der Gln. 5.5 und 5.6 auf Kernel-Machines und der Identifikation einer Funktion

$$\begin{aligned} R(s,a,s') &= Q(s,a) - \gamma Q(s', \pi(s')) && (6.19)\\ &= \sum_{i=1}^{l} \alpha_i (\delta_{a,a_i} K(s,s_i) - \gamma \delta_{\pi(s'),a_i} K(s',s_i)), && (6.20) \end{aligned}$$

die sich durch die Linearität der Faktoren α als

$$\begin{aligned} R(s,a,s') &= \sum_{i=1}^{l} \alpha_i \tilde{K}((s,a,s'),(s_i,a_i,s_i')) - b && (6.21)\\ \tilde{K}((s,a,s'),(\hat{s},\hat{a},\hat{s}')) &= \delta_{a,a_i} \hat{K}(s,s_i) - \gamma \delta_{\pi(s'),a_i} \hat{K}(s',s_i) && (6.22) \end{aligned}$$

schreiben lässt. Diese Rewards-Regression-Variante kann nun mit einer Vielzahl von Methoden gelöst werden. Wir haben exemplarisch in Abschn. 3.4 eine Reihe solcher Methoden vorgestellt. Bei der Ridge-Regression wird im Feature-Raum der Regularisierungsterm $\lambda \mathbf{w}^T \mathbf{w}$ zur Zielfunktion addiert.

Auf diese Weise ergibt sich mit $\tilde{\Phi} = \Phi(s) - \gamma \Phi(s')$ schließlich die Lösung des linearen Gleichungssystems

$$\begin{aligned} \left(\tilde{\Phi}\mathbf{w} - R\right)^2 + \lambda \mathbf{w}^T \mathbf{w} &= \min && (6.23)\\ \Leftrightarrow \tilde{\Phi}^T \left(\tilde{\Phi}\mathbf{w} - R\right) + \lambda \mathbf{w} &= 0 && (6.24)\\ \Leftrightarrow \left(\tilde{\Phi}^T \tilde{\Phi} + \lambda \mathbf{I}\right) \mathbf{w} &= \tilde{\Phi}^T R, && (6.25) \end{aligned}$$

was durch Kernelisierung zum linearen Gleichungssystem

$$\begin{aligned}\left(\tilde{K}\alpha - R\right)^2 + \lambda\alpha^T\tilde{K}\alpha &= \min &(6.26)\\ \Leftrightarrow \tilde{K}\left(\left(\tilde{K} + \lambda\mathbf{I}\right)\alpha - R\right) &= 0 &(6.27)\\ \Leftrightarrow \left(\tilde{K} + \lambda\mathbf{I}\right)\alpha &= R, &(6.28)\end{aligned}$$

führt.

Wenn es vom Kontext nicht gefordert wird, verzichten wir, insbesondere in der Matrixschreibweise, auf den Bias. Dieser kann als Teil des Kernels aufgefasst werden. Bei Support-Vector-Machines stellt sich dies anders dar, die implizite oder explizite Nutzung eines Bias verändert die angestrebte Lösung.

6.3 Bellman-Iteration und Bellman-Residuum

Die Kernel-Rewards-Regression erlaubt die Bestimmung der beiden Optimalitätskriterien Fixpunkt der Bellman-Iteration $Q = \Gamma TQ$, der der TD-Lösung entspricht und dem Minimum des Bellman-Residuums $\min_{Q\in H_Q} \|Q - TQ\|$. Hier ergibt sich das interessante Phänomen, dass die beiden Kriterien simultan erfüllt sind, sofern die Kernel-Matrix regulär ist und der Feature-Raum durch diese bestimmt wird. Dies ist (fast sicher) dann der Fall, wenn wir $\tilde{K}$ wie oben durch Addieren einer gewichteten Diagonalmatrix regularisieren.

Wie oben bereits ausgeführt, ist die Lösung von Gl. 6.28 zunächst der Fixpunkt der Bellman-Iteration wegen

$$\begin{aligned}K\alpha &= R + \gamma K'\alpha &(6.29)\\ \Leftrightarrow K\left(\tilde{K}\right)^{-1}R &= R + \gamma K'\left(\tilde{K}\right)^{-1}R &(6.30)\\ \Leftrightarrow (K - \gamma K')\left(\tilde{K}\right)^{-1}R &= \tilde{K}\left(\tilde{K}\right)^{-1}R &(6.31)\\ &= R. &(6.32)\end{aligned}$$

Sie ist aber außerdem das Minimum des (quadratischen) Bellman-Residuums $\|Q - TQ\| = \sqrt{(Q - TQ)^T(Q - TQ)}$. Da zur erwartungstreuen Schätzung des Bellman-Residuums für jede Beobachtung zwei verschiedene, unabhängig voneinander gesampelte Folgezustände mit den dabei vereinnahmten Rewards vorliegen müssen (siehe Abschn. 4.2 und 5.4), ergeben sich eine zusätzliche Kernel-Matrix für die alternativen Folgezustände K'' und ein zusätzlicher Reward-Vektor R'. Somit erhalten wir

$$R_{\text{Bellman}} = \left(K\alpha - \gamma K'\alpha - R\right)^T\left(K\alpha - \gamma K''\alpha - R'\right). \quad (6.33)$$

Dies ist aber bereits dann minimal, wenn o.B.d.A. das lineare Gleichungssystem aus Gl. 6.28 gelöst ist. Die alternative Beobachtung beeinflusst die Lösung also nicht (es handelt sich hier um den dritten Spezialfall aus Abschn. 5.4).

Diese Äquivalenz gilt im Allgemeinen nicht mehr, wenn wir uns von der Interpretation des regulären (oder regularisierten) Kernels als Skalarprodukt innerhalb des tatsächlichen Feature-Raumes lösen. Innerhalb des Feature-Raumes führt schließlich

$$\begin{aligned}R_{\text{Bellman}} &= \left(\Phi\mathbf{w} - \gamma\Phi'\mathbf{w} - R\right)^T\left(\Phi\mathbf{w} - \gamma\Phi''\mathbf{w} - R'\right) &(6.34)\\ &= \mathbf{w}^T\left(\Phi - \gamma\Phi'\right)^T\left(\Phi - \gamma\Phi''\right)\mathbf{w}\\ &\quad - \left((R')^T\left(\Phi - \gamma\Phi'\right) + R^T\left(\Phi - \gamma\Phi''\right)\right)\mathbf{w} + R^TR' = \min &(6.35)\\ \Leftrightarrow \mathbf{w} &= \left(\left(\Phi - \gamma\Phi'\right)^T\left(\Phi - \gamma\Phi''\right) + \left(\Phi - \gamma\Phi''\right)^T\left(\Phi - \gamma\Phi'\right)\right)^{-1}\\ &\quad \left((R')^T\left(\Phi - \gamma\Phi'\right) + R^T\left(\Phi - \gamma\Phi''\right)\right) &(6.36)\end{aligned}$$

zur Minimierung des Bellman-Residuums, was sich nicht mehr nur auf eine Beobachtung stützen lässt. Auf die alternative Beobachtung kann also nicht verzichtet werden, eine Voraussetzung, die im Batch-Learning-Fall in der Praxis nur in seltenen Fällen erfüllt werden kann.

6.4 Policy-Iteration in der Kernel-Rewards-Regression

In der oben vorgestellten Form kann die Regression bisher nur zu einer Q-Funktion für eine bestimmte, feste Policy führen, also Lösungen für das Policy-Evaluation-Problem bestimmen. Eine Möglichkeit für den Übergang zur Identifikation möglichst guter Policies besteht in der Anwendung einer deterministischen Policy-Iteration wie bei der LSPI, dies geschieht wie in Alg. 4 beschrieben.

6.4.1 Deterministische Policy-Iteration

Wir definieren eine Policy π in diesem Zusammenhang als optimal im starken Sinn, falls für den Schätzer Q und die Policy π auf den Beobachtungen

$$\forall i : \arg\max_{a'} Q(s'_i, a') \quad = \quad \pi(s'_i) \tag{6.37}$$

gilt, also das Maximum der Q-Funktion an den beobachteten Folgezuständen tatsächlich der Policy entspricht, die das Policy-Evaluation-Problem für π löst. Man beachte dabei, dass Q nur ein Schätzer für die wahre Q-Funktion ist, denn sonst wäre die Bedingung selbstverständlich stets erfüllt.

Dementsprechend ist die Wahl der jeweils besten Aktion in jeder Iteration eine naheliegende Option (siehe Abschn. 2.2.2). In der Praxis führt dieses Vorgehen zu sehr schneller Konvergenz für eine Vielzahl von RL-Problemen [57], meistens sind nur wenige, etwa vier oder fünf Iterationen erforderlich. In anderen Fällen konvergiert das Verfahren nicht und dann ist möglicherweise keine stark optimale Policy für die gegebenen Beobachtungen und Parameter auffindbar. Beim KRR hängt dies jedoch sehr von der Wahl des Kernels ab.

So besteht etwa die Möglichkeit, Bedingungen an den Kernel und die Beobachtungen zu stellen, so dass Konvergenz garantiert werden kann. Eine mögliche Bedingung besteht darin, dass im Wesentlichen für jede Beobachtung i eine benachbarte Beobachtung j existieren muss, die innerhalb des Feature-Raumes nur einen bestimmten Maximalabstand zur ersten Beobachtung aufweisen darf. Diese Bedingungen sind jedoch nur selten anwendbar und die Annahmen in der Praxis schwer zu überprüfen. Daher verzichten wir an dieser Stelle auf detailliertere Ausführungen dazu.

Berücksichtigt man zusätzlich eine Zufallskomponente, so dass nicht zwangsläufig die lokal beste Aktion, sondern mit einer zuvor festgelegten Wahrscheinlichkeit eine zufällige Aktion ausgewählt wird, dann kann nach endlich vielen Iterationen stets Konvergenz gegen eine stark optimale Policy garantiert werden, wenn sie existiert.

6.4.2 Stochastische Policy-Iteration

Als weitere Alternative betrachten wir stochastische Policies, deren Anwendung, unabhängig vom gewählten Kernel und den Beobachtungen, stets Konvergenz ermöglichen, auch wenn keine stark optimale Policy existiert. Dazu definieren wir eine stochastische Policy π als schwach optimal, falls für den resultierenden Schätzer Q der Q-Funktion und alle Beobachtungen der Zusammenhang

$$\forall i : \tilde{A}_i \quad = \quad \arg\max_{a'} Q(s'_i, a') \tag{6.38}$$

$$\forall a : \pi(s_i, a) > 0 \quad \Leftrightarrow \quad a \in \tilde{A}_i \tag{6.39}$$

Algorithmus 4 Kernel-Rewards-Regression-Algorithmus

Bedingungen: gegeben eine Menge von Transitionen $\{(s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l)\}$, ein Kernel K, Regressionsparameter (z.B. C und ε), Diskontierungsfaktor $0 < \gamma < 1$ und ein Zufalls-Faktor $0 \leq \rho \leq 1$

Zusicherungen: bestimmt eine optimale Policy im starken Sinn

setze $\forall i : \pi^0(s'_i)$ zufällig unter den Nebenbedingungen $\forall i : 1 \leq \pi^0(s'_i) \leq |A|$

setze $t = 0$

solange die gewünschte Genauigkeit nicht erreicht ist **führe aus**

setze $t = t + 1$

setze $\forall i, j : \tilde{K}_{i,j} = K((s_i, a_i), (s_j, a_j)) - \gamma K((s'_i, \pi^{t-1}(s'_i)), (s_j, a_j))$

löse das Regressionsproblem $\tilde{K}\alpha = R$ bzgl. der Regressionsparameter (z.B. C und ε)

setze $\forall i : \pi^t(s'_i) = \arg\max_{a'} \sum_{j=1}^{l} K((s'_i, a'), (s_j, a_j))$

wähle $\pi^t(s'_i)$ zufällig mit Wahrscheinlichkeit ρ

wenn ein Attraktor erreicht wurde **dann**

wähle π^t, für das $\sum_{i=1}^{l} \sum_{j=1}^{l} K((s_i, a_i), (s_j, a_j)) = \max$ und verlasse die Schleife

ende wenn

ende solange

setze $\forall s \in S : \pi(s) = \arg\max_{a'} \sum_{i=1}^{l} \alpha_i K((s, a'), (s_i, a_i))$

gebe π zurück

gilt. Mit anderen Worten, für jeden Zustand muss für jede Aktion mit positiver Wahrscheinlichkeit der entsprechende Q-Wert maximal sein. Für stochastische Policies verändert sich dann das Regressionsproblem zu

$$R = \left(K - \gamma \sum_{i=1}^{|A|} D_i K'_i \right) \alpha, \tag{6.40}$$

wobei D_i Diagonalmatrizen mit ausschließlich positiven Einträgen $(D_i)_{j,j} = \pi(s_i, a_j)$ und demnach $\sum_{i=1}^{|A|} D_i = \mathbf{I}$ sowie K'_a jeweils die Kernels der Folgezustände unter der Annahme, dass in jedem Zustand Aktion a ausgeführt wird, sind. Wir definieren dazu

$$\pi^{t+1}(s_i, a') = \begin{cases} \pi^t(s_i, a') + d_i & : \quad a_i = a' \\ \pi^t(s_i, a') \left(\frac{1 - (\pi^t(s_i, a') + d_i)}{1 - \pi^t(s_i, a')} \right) & : \quad \text{sonst} \end{cases} \tag{6.41}$$

$$d_i = \min \left(1 - \pi^t(s_i, a_i), \frac{1}{t} \right). \tag{6.42}$$

Dies soll schwache Optimalität gewährleisten. Alg. 5 fasst die Verfahrensweise zusammen. In jeder Iteration überprüft der Algorithmus, welche Folgeaktion optimal im deterministischen Sinn wäre und verändert die stochastische Policy zu Gunsten dieser Aktion.

Wie man leicht einsieht, führt dieses Vorgehen zu einer Änderung der stochastischen Policies mit harmonischer Änderungsrate $O(1/t)$, so dass einerseits die Policy konvergieren muss und andererseits jede stochastische Policy erreicht werden kann, da die harmonische Reihe divergiert. Im Limit muss für jeden Zustand s und für jede Aktion a, für die $\pi(s, a) > 0$ ist, $Q(s, a) = \max_{a'} Q(s, a')$ den dort maximal möglichen Wert annehmen.

6.5 Support-Vector-Rewards-Regression

Die Anwendung von Support-Vector-Machines für die KRR erfordert etwas mehr Aufwand, da der Kernel $\tilde{K}$ im Allgemeinen nicht symmetrisch und positiv definit ist. Es gibt zwei

Algorithmus 5 Stochastischer Kernel-Rewards-Regression-Algorithmus

Bedingungen: gegeben eine Menge von Transitionen $\{(s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l)\}$, ein Kernel K, Regressionsparameter (z.B. C und ε) und ein Diskontierungsfaktor $0 < \gamma < 1$

Zusicherungen: bestimmt eine optimale deterministische Policy im schwachen Sinn

setze $\forall i, j : \pi^0(s'_i, a_j) = \frac{1}{|A|}$

setze $t = 0$

solange die gewünschte Genauigkeit nicht erreicht ist **führe aus**

setze $t = t + 1$

setze $\forall i, j : \tilde{K}_{i,j} = K((s_i, a_i), (s_j, a_j)) - \gamma \sum_{a \in A} \pi^{t-1}(s'_i, a) K((s'_i, j), (s_j, a_j))$

löse das Regressionsproblem $\tilde{K}\alpha = \mathbf{y}$ bzgl. der Regressionsparameter (z.B. C und ε)

setze $\forall i : a_{i,\max} = \arg\max_{a'} \sum_{j=1}^{l} \alpha_i K((s'_i, a'), (s_j, a_j))$

setze $\forall i : d_i = \min\left(\frac{1}{t}, 1 - \pi^{t-1}(s_i, a_{i,\max})\right)$

setze $\forall i : \pi^t(s_i, a_{i,\max}) = \pi^{t-1}(s_i, a_{i,\max}) + d_i$

setze $\forall i : \forall a_j \neq a_{i,\max} : \pi^t(s_i, a_j) = \frac{1-\pi^t(s_i, a_{i,\max})}{1-\pi^t(s_i, a_{i,\max})+d_i} \pi^{t-1}(s_i, a_j)$

ende solange

setze $\pi(s) = \arg\max_{a'} \sum_{i=1}^{l} \alpha_i K((s, a'), (s_i, a_i))$

gebe π zurück

Möglichkeiten, dieses Problem zu beheben. Für die Kernelisierung $\mathbf{w} = \sum_{i=1}^{l} \alpha_i K((s_i, a_i), \cdot)$ minimiert man

$$\mathbf{w}^T \mathbf{w} = \alpha^T \left(K + \frac{1}{C}\mathbf{I}\right) \alpha = \min \tag{6.43}$$

unter den Nebenbedingungen, dass

$$\left| \left(\tilde{K} + \frac{1}{C}\mathbf{I}\right) \alpha - R \right| \leq \varepsilon. \tag{6.44}$$

Bei diesem naheliegenden Ansatz ist K symmetrisch und positiv definit per Konstruktion, falls $\hat{K}$ symmetrisch und positiv definit ist. Dabei entspricht Gl. 6.43 der Zielfunktion analog zu Gl. 3.64 und Gl. 6.44 der ε-Schlauch-Bedingung analog zu Gl. 3.70. Diese Technik hat allerdings den Nachteil, dass die Kernel-Matrizen in der Zielfunktion und in den Nebenbedingungen verschieden sind. Aus diesem Grund ist das vorliegende quadratische Optimierungsproblem zwar prinzipiell lösbar, jedoch lassen sich für die SVM spezialisierte Algorithmen [74, 95] nicht ohne Weiteres anwenden.

Eine weitere Möglichkeit besteht in einer alternativen Kernelisierung. Durch die Substitution

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i (K((s_i, a_i), \cdot) - \gamma K((s'_i, a'_i), \cdot)) \tag{6.45}$$

erhält man das Regressionsproblem

$$R = (K - \gamma K'')\alpha - \gamma(K' - \gamma K''')\alpha \tag{6.46}$$

mit K wie oben, eventuell regularisiert und

$$K'_{i,j} = K((s_i, a_i), (s'_j, a'_j)) \tag{6.47}$$

$$K''_{i,j} = K((s'_i, a'_i), (s_j, a_j)) \tag{6.48}$$

$$K'''_{i,j} = K((s'_i, a'_i), (s'_j, a'_j)). \tag{6.49}$$

Das ergibt sich unmittelbar aus der Tatsache, dass die Stützstellen nun Linearkombinationen von Zustand und Folgezustand sind. Durch Aufaddieren erhält man unmittelbar den neuen Kernel

$$K^{\mathrm{qp}} = K - \gamma(K' + K'') + \gamma^2 K''', \tag{6.50}$$

der tatsächlich per Konstruktion symmetrisch und positiv definit ist. Hier können Algorithmen für die SVM also zum Einsatz kommen. Die Q-Funktion ergibt sich schließlich zu

$$Q(s,a) = \sum_{i=1}^{l} \alpha_i \left(K((s_i,a_i),(s,a)) - \gamma K((s_i',a_i'),(s,a))\right) - b_a \tag{6.51}$$

mit angemessenem Bias b_a. Der Bias muss auch hier stets so gewählt werden, dass die Nebenbedingungen möglichst großzügig erfüllt werden (siehe auch Abschn. 3.4.2). Wir erhalten also ein klar definiertes Support-Vector-Regression-Problem.

6.6 Direkte Policy-Identifikation – quadratische Programmierung für die Explicit-Kernel-Rewards-Regression

Schließlich präsentieren wir einen Ansatz zur Lösung von Optimalregelungsproblemen ohne eine explizite Policy-Iteration, vielmehr wird die Lösung mit einem direkten Verfahren bestimmt und als ein quadratisches Optimierungsproblem formuliert. Dies hat u.a. den Vorteil, dass keine besonderen Vorkehrungen bzgl. der Konvergenz getroffen werden müssen. Diese Methode, die wir Explicit-Kernel-Rewards-Regression (EKRR) nennen, hat daher mehr Ähnlichkeit mit dem Supervised-Learning und ist, wie auch die Neural-Rewards-Regression (siehe Kap. 7), ein direktes Lösungsverfahren.

Der Einfachheit halber konzentrieren wir uns zunächst auf zwei diskrete Aktionen, die Verallgemeinerung zu einer beliebigen endlichen Anzahl von Aktionen ist jedoch ohne Weiteres möglich, wir werden wieder darauf zurückkommen.

Wir beginnen mit der Formulierung der Bedingungen

$$R = K\alpha - \gamma \max\left(K_1'\alpha, K_2'\alpha\right) \tag{6.52}$$

$$= \min\left((K - \gamma K_1')\alpha, (K - \gamma K_2')\alpha\right). \tag{6.53}$$

Dabei sind K_a' jeweils die Kernels der Folgezustände, unter der Annahme, dass in jedem Zustand Aktion a ausgeführt wird, die max- und min-Operationen sind hier komponentenweise aufzufassen. In den Folgezuständen werden also per Konstruktion stets die optimalen Aktionen ausgeführt und Gl. 6.52 repräsentiert die Kernel-Rewards-Regression-Formulierung für die Bellman-Optimalitäts-Gleichung. Um diese Bedingungen in reguläre lineare Bedingungen zu überführen, substituieren wir

$$\mathbf{v}_+ - \mathbf{v}_- = \gamma(K_2' - K_1')\alpha \tag{6.54}$$

mit den zusätzlichen Nebenbedingen

$$\mathbf{v}_+ \geq 0 \tag{6.55}$$

$$\mathbf{v}_- \geq 0. \tag{6.56}$$

Dabei repräsentiert $\mathbf{v}_+$ komponentenweise die positive Differenz von $\gamma K_2'\alpha$ zu $\gamma K_1'\alpha$ und $\mathbf{v}_-$ symmetrisch dazu die positive Differenz von $\gamma K_1'\alpha$ zu $\gamma K_2'\alpha$. Die Umformulierung von Gl. 6.52 nach

$$\left(K - \frac{\gamma}{2}\left(K_1' + K_2'\right)\right)\alpha - \frac{\mathbf{v}_+ + \mathbf{v}_-}{2} = R \tag{6.57}$$

ist daher möglich, wenn zusätzlich die quadratischen Nebenbedingungen

$$\forall i : \mathbf{v}_{i,+}\mathbf{v}_{i,-} = 0 \tag{6.58}$$

erfüllt sind. Denn es muss mindestens $\mathbf{v}_{i,+} = 0$ oder $\mathbf{v}_{i,-} = 0$ für alle i gelten, da nur eine einseitige positive Differenz betrachtet werden kann. Ist etwa o.B.d.A. $\mathbf{v}_{1,+} > 0$ und $\mathbf{v}_{1,-} = 0$, dann ergibt sich

$$\begin{aligned} R_1 &= \left(K_{1,\cdot} - \frac{\gamma}{2}((K'_1)_{1,\cdot} + (K'_2)_{1,\cdot})\right)\alpha - \frac{\mathbf{v}_{1,+} + \mathbf{v}_{1,-}}{2} && (6.59)\\ &= \left(K_{1,\cdot} - \frac{\gamma}{2}((K'_1)_{1,\cdot} + (K'_2)_{1,\cdot})\right)\alpha - \frac{\gamma(K'_2)_{1,\cdot}\alpha - \gamma(K'_1)_{1,\cdot}\alpha}{2} && (6.60)\\ &= (K_{1,\cdot} - \gamma(K'_2)_{1,\cdot})\,\alpha, && (6.61) \end{aligned}$$

was gerade dem gewünschten Ausdruck für diese Komponente entspricht, da die Q-Funktion für die zweite Aktion, repräsentiert durch $K'_2\alpha$, einen höheren Wert liefert.

Quadratische Programmierung sieht jedoch in dieser Standard-Formulierung keine quadratischen Nebenbedingungen vor. Fassen wir stattdessen Gl. 6.58 als Ungleichungsnebenbedingung

$$\forall i : \mathbf{v}_{i,+}\mathbf{v}_{i,-} \leq s \tag{6.62}$$

auf, dann kann die Wahl eines angemessenen $s \in \mathbb{R}$ stets in ein Minimierungsproblem mit geeignetem Vorfaktor überführt werden, der von s abhängt. Da dies jedoch bisher das einzige Optimierungskriterium ist, minimieren wir schließlich

$$\sum_{i=1}^{l} \mathbf{v}_{i,+}\mathbf{v}_{i,-} = \min. \tag{6.63}$$

Also kann das quadratische Optimierungsproblem als

$$\begin{aligned} \begin{pmatrix} \mathbf{v}_+^T & \mathbf{v}_-^T \end{pmatrix} \begin{pmatrix} 0 & \mathbf{I} \\ \mathbf{I} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} &= \min && (6.64)\\ \begin{pmatrix} K - \frac{\gamma}{2}(K'_1 + K'_2) & -\frac{1}{2}\mathbf{I} & -\frac{1}{2}\mathbf{I} \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} &= R && (6.65)\\ \begin{pmatrix} \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} &\geq 0 && (6.66)\\ \begin{pmatrix} \gamma(K'_2 - K'_1) & -\mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} &= 0. && (6.67) \end{aligned}$$

geschrieben werden. Innerhalb des gültigen Lösungsbereiches dieses quadratischen Programms ist $\begin{pmatrix} (\mathbf{v}_+)^T & (\mathbf{v}_-)^T \end{pmatrix} \geq 0$ genau dann mit Gleichheit erfüllt, wenn eine Lösung des Optimalregelungsproblems im starken Sinn existiert.

Für mehr als zwei verschiedene Aktionen können wir die Nebenbedingungen rekursiv definieren und erhalten

$$\begin{aligned} \mathbf{m}_1 &= (K - \gamma K'_1)\,\alpha && (6.68)\\ \mathbf{m}_i &= \min\left((K - \gamma K'_i)\,\alpha, \mathbf{m}_{i-1}\right) && (6.69)\\ &= \frac{(K - \gamma K'_i)\,\alpha - \mathbf{m}_{i-1}}{2} - \frac{\mathbf{v}_+^i + \mathbf{v}_-^i}{2} && (6.70)\\ \mathbf{m}_{|A|} &= R && (6.71)\\ \mathbf{v}_+^1 &= \mathbf{v}_-^1 = 0 && (6.72)\\ \mathbf{v}_+^i - \mathbf{v}_-^i &= (K - \gamma K'_i)\,\alpha - \mathbf{m}_{i-1}, && (6.73)\\ & \mathbf{v}_+^i \geq 0, \mathbf{v}_-^i \geq 0, \mathbf{v}_+^i\mathbf{v}_-^i = 0. && (6.74) \end{aligned}$$

Algorithmus 6 Explicit-Kernel-Rewards-Regression-Algorithmus

Bedingungen: gegebene Menge von Transitionen $T = \{(s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l)\}$ mit $|A| = 2$, ein Kernel K, Regressionsparameter (z.B. C und ε), Diskontierungsfaktor $0 < \gamma < 1$ sowie angemessene quadratische und lineare Regularisierungsoperatoren Ω und ω

Zusicherungen: bestimmt eine optimale Policy im starken Sinn
setze $\forall i, j : K_{i,j} = K((s_i, a_i), (s_j, a_j))$
setze $\forall i, j : \forall a' \in A : (K'_{a'})_{i,j} = K((s'_i, a'), (s_j, a_j))$
löse das quadratische Optimierungsproblem gegeben durch

$$\mathbf{v}_+^T \mathbf{v}_- + \alpha^T \Omega \alpha + \omega^T \alpha = \min \tag{6.75}$$

$$\begin{pmatrix} K - \frac{\gamma}{2}(K'_1 + K'_2) & -\frac{1}{2}\mathbf{I} & -\frac{1}{2}\mathbf{I} \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} = R \tag{6.76}$$

$$\begin{pmatrix} \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} \geq 0 \tag{6.77}$$

$$\begin{pmatrix} \gamma(K'_2 - K'_1) & -\mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} = 0 \tag{6.78}$$

unter den eventuellen Nebenbedingungen

$$\begin{pmatrix} K - \frac{\gamma}{2}(K'_1 + K'_2) & -\frac{1}{2}\mathbf{I} & -\frac{1}{2}\mathbf{I} \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} \leq R + \varepsilon \tag{6.79}$$

$$\begin{pmatrix} K - \frac{\gamma}{2}(K'_1 + K'_2) & -\frac{1}{2}\mathbf{I} & -\frac{1}{2}\mathbf{I} \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} \geq R - \varepsilon, \tag{6.80}$$

die Gl. 6.76 ersetzen
setze $\forall s \in S : \pi(s) = \arg\max_{a'} \sum_{i=1}^{l} \alpha_i K((s, a'), (s_i, a_i))$
gebe π zurück

6.6.1 Regularisierung und weitere Bedingungen

Da wir nun bereits ein geeignetes quadratisches Programm konstruiert haben, können wir auch entsprechende Regularisierungstechniken, die in Kernel-Machines und besonders in der Support-Vector-Machine verwendet werden, zum Einsatz bringen. Es ist ferner möglich, weitere problemangepasste lineare Nebenbedingungen oder lineare sowie quadratische Komponenten der Zielfunktion hinzuzufügen. Wir betrachten z.B. die erweiterte Zielfunktion

$$\begin{pmatrix} \alpha^T & \mathbf{v}_+^T & \mathbf{v}_-^T \end{pmatrix} \begin{pmatrix} c_1 K & 0 & 0 \\ 0 & 0 & \mathbf{I} \\ 0 & \mathbf{I} & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} = \min \tag{6.81}$$

mit den zusätzlichen Nebenbedingungen

$$\begin{pmatrix} K - \frac{\gamma}{2}(K'_1 + K'_2) & -\frac{1}{2}\mathbf{I} & -\frac{1}{2}\mathbf{I} \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} \leq R + \varepsilon \tag{6.82}$$

$$\begin{pmatrix} K - \frac{\gamma}{2}(K'_1 + K'_2) & -\frac{1}{2}\mathbf{I} & -\frac{1}{2}\mathbf{I} \end{pmatrix} \begin{pmatrix} \alpha \\ \mathbf{v}_+ \\ \mathbf{v}_- \end{pmatrix} \geq R - \varepsilon, \tag{6.83}$$

die die Gleichheitsbedingungen aus Gl. 6.57 ersetzen.

Diese Umgestaltung des KRR kann nun als Alternative zur Support-Vector-Rewards-Regression verwendet werden, da Regularisierungstechniken der SVM zur Anwendung kommen. Der zusätzliche Term $c_1 \alpha^T K \alpha$ etwa entspricht gerade der quadrierten Länge des Gewichtsvektors im Feature-Raum.

6.6.2 Konvergenz und Optimalität

Bedauerlicherweise ist das so entstandene quadratische Programm nicht konvex. Dennoch kann man von der Theorie der nicht-konvexen Optimierung profitieren [90, 14], um explizit dateneffiziente Policies zu ermitteln.

Falls keine stark optimale Policy existiert, ist EKRR nicht in der Lage, schwach optimale Policies zu bestimmen, da nur deterministische Policies miteinander verglichen werden. Da das Einhalten der Nebenbedingungen in Gl. 6.65 wegen der Unterbestimmtheit des Gleichungssystems fast sicher möglich ist, wird auch stets eine Lösung des Optimierungsproblems zu finden sein. Dann ist o.B.d.A. mindestens $\mathbf{v}_{1,+} > 0$ und $\mathbf{v}_{1,-} > 0$ und wir erhalten

$$
\begin{aligned}
R_1 &= \left(K_{1,\cdot} - \frac{\gamma}{2}\left((K_1')_{1,\cdot} + (K_2')_{1,\cdot}\right)\right)\alpha - \frac{\mathbf{v}_{1,+} + \mathbf{v}_{1,-}}{2} && (6.84)\\
&= \left(K_{1,\cdot} - \frac{\gamma}{2}\left((K_1')_{1,\cdot} + (K_2')_{1,\cdot}\right)\right)\alpha && (6.85)\\
&\quad - \frac{\gamma\,(K_2')_{1,\cdot}\,\alpha - \gamma\,(K_1')_{1,\cdot}\,\alpha + 2\mathbf{v}_{1,-}}{2} && (6.86)\\
&= \left(K_{1,\cdot} - \gamma\,(K_2')_{1,\cdot}\right)\alpha - \mathbf{v}_{1,-} && (6.87)\\
&= \left(K_{1,\cdot} - \frac{\gamma}{2}\left((K_1')_{1,\cdot} + (K_2')_{1,\cdot}\right)\right)\alpha && (6.88)\\
&\quad - \frac{\gamma\,(K_1')_{1,\cdot}\,\alpha - \gamma\,(K_2')_{1,\cdot}\,\alpha + 2\mathbf{v}_{1,+}}{2} && (6.89)\\
&= \left(K_{1,\cdot} - \gamma\,(K_1')_{1,\cdot}\right)\alpha - \mathbf{v}_{1,+}. && (6.90)
\end{aligned}
$$

Die Fehlerterme $\mathbf{v}_{1,+}$ und $\mathbf{v}_{1,-}$ führen jeweils zu einer nicht korrekten Wiedergabe der Bellman-Optimalitäts-Gleichung durch ein Überschätzen der Rewards, was letztlich auch ein Überschätzen der Q-Funktion zur Folge hat. Die Policy ergibt sich aber nach wie vor direkt aus der Q-Funktion.

6.7 Kontinuierliche Aktionen

Kernel-Rewards-Regression erlaubt in begrenztem Umfang auch den Einsatz kontinuierlicher Aktionen. Dies beruht auf der Idee, den Kernel so zu wählen, dass die Maximalaktionen für jede mögliche Q-Funktion Q mit $\alpha > 0$ die beobachtete Aktion an einer der Stützstellen ist. Da alle Beobachtungen Stützstellen sind, ist also jede beobachtete Aktion ein Kandidat.

Sei dazu $K(\mathbf{x}, \mathbf{z}) = K(\|\mathbf{x} - \mathbf{z}\|)$, stückweise zweimal differenzierbar und $\frac{\partial^2 K(c)}{\partial c^2} > 0$. Dann muss für alle $\alpha > 0$ und Stützstellen $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ sowie $f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$ der Zusammenhang

$$
\arg\max_{\mathbf{x}} f(\mathbf{x}) \cup X \quad = \quad X \tag{6.91}
$$

gelten. Dies kann man einsehen, indem man das Gegenteil der Behauptung betrachtet, einige der Nicht-Stützstellen seien maximal, mindestens ein $\mathbf{x} \notin X$. Wir definieren $f' = \left(\frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}) \to \mathbf{x}\right)$. Im ersten Fall ist $f'(\mathbf{x}) \neq 0$. Dann kann jedoch $f(\mathbf{x})$ nicht maximal sein. Im zweiten Fall gilt $f'(\mathbf{x}) = 0$. Dann wählen wir einen Vektor $\mathbf{z}$, so dass wenigstens ein i existiert mit $(\mathbf{x}_i - \mathbf{x})^T \mathbf{z} > 0$. In Richtung $\mathbf{z}$ erhöhen sich dann jedoch alle Gradientenbeiträge

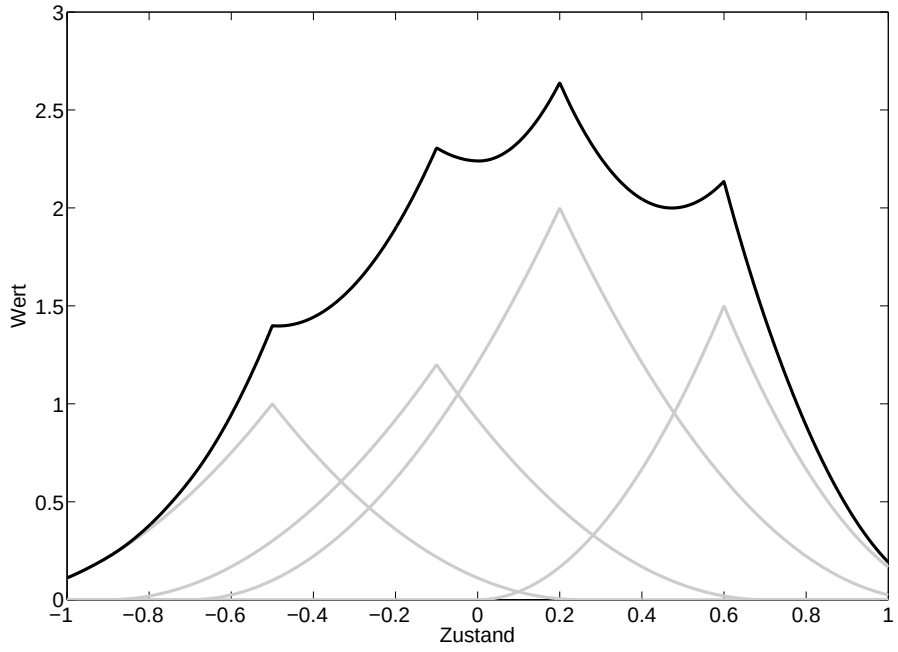


Abbildung 6.1: Kernels für den Umgang mit kontinuierlichen Aktionen. Fällt die erste Ableitung des Abstandes zum Kernel-Center $K(\|\mathbf{x}-\mathbf{z}\|)$ monoton (Kernels in grauen Linien), dann kann sich in der Umgebung eines Kernel-Centers kein lokales Maximum einer Linearkombination von Kernels (schwarze Linie) befinden außer dem Kernel-Center selbst oder dem eines anderen Kernels. Daher muss auch das globale Maximum an einem der Kernel-Center liegen.

derjenigen $\mathbf{x}_j$, für die $(\mathbf{x}_j - \mathbf{x})^T\mathbf{z} > 0$ (insbesondere auch für $j = i$), für alle anderen verringern sich die Gradientenbeiträge. Da diese aber in $\mathbf{x}$ in Richtung $\mathbf{z}$ ausgeglichen waren, gilt für ein hinreichend kleines ε der Zusammenhang $f'(\mathbf{x} + \varepsilon\mathbf{z})^T\mathbf{z} > 0$, was zu einem Anstieg von f in Richtung $\mathbf{z}$ führt. Daher muss $\mathbf{x}$ ein Minimum oder Sattelpunkt sein und $f(\mathbf{x})$ ist nicht maximal.

Dann kann bei der Policy-Iteration für jede Q-Funktion mit quadratischem Aufwand die Maximalaktion für jeden Folgezustand s'_i bestimmt werden, falls sich der Kernel K multiplikativ aus einem beliebigen positiv definiten Kernel K_S und einem Kernel K_A, der o.g. Eigenschaften erfüllt, zusammensetzt, also

$$K((s,a),(\hat{s},\hat{a})) \quad = \quad K_S(s,\hat{s})K_A(a,\hat{a}). \tag{6.92}$$

Dazu muss lediglich für alle bisher beobachteten Aktionen a_j der Wert

$$Q(s'_i, a_j) \quad = \quad \sum_{k=1}^{l} \alpha_k K((s'_i, a_j),(s_k, a_k)) \tag{6.93}$$

$$= \quad \sum_{k=1}^{l} \alpha_k K_S(s'_i, s_k) K_A(a_j, a_k) \tag{6.94}$$

bestimmt werden, da $K_S(s'_i, s_k)$ nur von i und k abhängt und daher die Substitution $\hat{\alpha}_k = \alpha_k K_S(s'_i, s_k)$ für festes i möglich ist, also $Q(s'_i, \cdot)$ nur an den Stützstellen a_j ein Maximum annehmen kann.

6.8 Intrinsic-Recurrent-Support-Vector-Machine für Systeme höherer Ordnung

Das Konzept der sogenannten Intrinsic-Recurrent-Support-Vector-Machine (IRSVM) kombiniert Kernel-Regression mit dynamischen Systemen und ist angelehnt an Arbeiten zum Reservoir-Computing [48, 103]. Im Reservoir-Computing wird ein dynamischer Feature-Raum erzeugt, der vergangene Beobachtungen miteinbezieht. Dieser kann dann als Kernel-Machine interpretiert werden, jedoch ist der Feature-Raum explizit durch das sogenannte Reservoir gegeben. Die Transformation des Reservoirs von einem Zeitschritt zum nächsten erfolgt nach Regeln, die einer erheblichen Zufallskomponente unterliegen. Bei der IRSVM hingegen wird dieser Übergang adaptiert und der Feature-Raum ist nur implizit durch den Kernel gegeben.

Im Vergleich zur statischen SVM führen wir zusätzlich einen internen Zustand $\mathbf{x}$ und einen entsprechenden Gewichtsvektor $\hat{\mathbf{w}}$ ein. Der externe Zustand (des Zustandsraumes im Reinforcement-Learning) wird hier mit $\mathbf{s}$ bezeichnet. Die Regressionsfunktion, die schließlich auch die Q-Funktion repräsentieren kann, erhält dann die Gestalt

$$f(\mathbf{s},\mathbf{x}) \quad = \quad \mathbf{w}^T\mathbf{s} + \hat{\mathbf{w}}^T\mathbf{x} + b. \tag{6.95}$$

Die zeitliche Entwicklung des internen Zustandes erfolgt durch Matrizen A und B, so dass $\mathbf{x}' = A\mathbf{x} + B\mathbf{s}$. In [97] wird die Funktionsweise der IRSVM für Klassifikationsprobleme hergeleitet und erprobt. Bei Regressionsproblemen ist von einem primären Optimierungsproblem

$$\begin{aligned} \rho\left(\mathbf{w},\hat{\mathbf{w}},A,B\right) \quad &= \quad \frac{1}{2}\Bigg(\mathbf{w}^T\mathbf{w} + \hat{\mathbf{w}}^T\hat{\mathbf{w}} + \mathbf{Tr}\left(AA^T\right) + \mathbf{Tr}\left(BB^T\right) \\ &\qquad + \sum_{i=1}^{d}\left(\xi_i^2 + \hat{\xi}_i^2\right)\Bigg) = \min_{\mathbf{w},\hat{\mathbf{w}},A,B} \end{aligned} \tag{6.96}$$

$$\text{mit} \quad y_i - \left(\mathbf{w}^T\mathbf{s}_i + \hat{\mathbf{w}}^T\mathbf{x}_i + b\right) \quad \leq \quad \varepsilon + \xi_i \quad \forall i \tag{6.97}$$

$$\left(\mathbf{w}^T\mathbf{s}_i + \hat{\mathbf{w}}^T\mathbf{x}_i + b\right) - y_i \quad \leq \quad \varepsilon + \hat{\xi}_i \quad \forall i \tag{6.98}$$

$$\frac{A\mathbf{x}_{i-1} + B\mathbf{s}_{i-1}}{\|A\mathbf{x}_{i-1} + B\mathbf{s}_{i-1}\|} \quad = \quad \mathbf{s}_i \quad \forall i \tag{6.99}$$

auszugehen. Mit Hilfe des Lagrange-Formalismus konstruiert man das entsprechende duale Problem, das dann mit quadratischer Programmierung gelöst werden kann. Es ist zu beachten, dass außer ρ auch andere Regularisierungsterme zur Anwendung kommen können, etwa $2\rho' = \mathbf{w}^T\mathbf{w} + \hat{\mathbf{w}}^T\hat{\mathbf{w}}$, bei dem A und B nicht bestraft werden. Die Normierung von $\mathbf{s}_i$ ist eine einfache Möglichkeit, Divergenz des Verfahrens zu vermeiden. Daher sind keine besonderen Anforderungen an die Operatoren A und B zu stellen.

Um die kernelisierte Variante zu erhalten, wird nun das Skalarprodukt $\mathbf{x}^T\mathbf{z}$ durch einen symmetrischen und positiv definiten Kernel ersetzt, so dass

$$\mathbf{w}^T\mathbf{s} \quad = \quad \sum_{i=1}^{T}\alpha_i y_i K(\mathbf{s}_i,\mathbf{s}) \tag{6.100}$$

$$\hat{\mathbf{w}}^T\mathbf{x} \quad = \quad \sum_{i=1}^{T}\alpha_i y_i K'\left(\mathbf{x}_i\left(A,B\right),\mathbf{s}\right), \tag{6.101}$$

wobei $K(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$ und $K'(\mathbf{s}_i,\mathbf{s}_j) = \Phi'(\mathbf{s}_i)^T\Phi'(\mathbf{s}_j)$ mit Φ und Φ' Abbildungen, die in zwei verschiedene Feature-Räume transformieren, gesetzt werden.

Wie in [97] für den Klassifikationsfall ausgeführt, wird man auch im Regressionsfall einen Algorithmus verwenden können, der alternierend quadratische Programmierung und Gradientenabstieg nutzt. Schließlich kann die Technik, angelehnt an die Support-Vector-Regression (siehe Abschn. 3.4), auch für die Rewards-Regression zum Einsatz gebracht werden.

Kapitel 7

Neural-Rewards-Regression

Die Neural-Rewards-Regression ist die Umsetzung der Rewards-Regression mit neuronalen Netzen. Während bei der Kernel-Rewards-Regression sich der Hypothesenraum für die Reward-Funktion mit Hilfe der Linearität der Kernel-Komponenten aufbauen lässt, beruht die Realisierbarkeit für neuronale Netze im Konzept der Shared-Weights, so dass eine Regressionsaufgabe an die Rewards gestellt wird, die eine Extraktion der Q-Funktion ermöglicht.

Für diskrete Aktionen wird die Q-Funktion entweder parallel in $|A|$ verschiedenen neuronalen Netzen

$$Q(s,a) \quad = \quad N_a(s) \tag{7.1}$$

ausgedrückt, wobei die Q-Funktionen für jede Aktion getrennt sind oder als ein monolithisches Netz

$$Q(s,a) \quad = \quad N(s,a) \tag{7.2}$$

realisiert. Die Alternativen unterscheiden sich in der Generalisierungsfähigkeit über den Aktionsraum. Die zu bestimmende Reward-Funktion kann dann die Gestalt

$$R(s,a,s') \quad = \quad N_a(s) - \gamma \max_{a'} N_{a'}(s') \tag{7.3}$$

oder

$$R(s,a,s') \quad = \quad N(s,a) - \gamma \max_{a'} N(s',a') \tag{7.4}$$

annehmen. Dieses Regressionsproblem kann mittels Gradientenabstieg (siehe Abschn. 3.4) gelöst werden. Dies führt zum Minimieren des Bellman-Residuums

$$L \quad = \quad \sum_{i=1}^{l} F(L_i) = \sum_{i=1}^{l} F\left(Q(s_i,a_i) - \gamma V(s_i') - r_i\right) \tag{7.5}$$

mit θ als den Parametern des neuronalen Netzes und F einer angemessenen Fehlerfunktion. Man beachte dabei, dass hier das von uns so bezeichnete deterministische Bellman-Residuum betrachtet wird, das kein erwartungstreuer Schätzer für das tatsächliche Bellman-Residuum für stochastische Probleme ist (siehe Abschn. 5.4). Der Gradient

$$\frac{\partial L}{\partial \theta} \quad = \quad \sum_{i=1}^{l} F'(L_i) \frac{\partial}{\partial \theta} \left(Q(s_i,a_i) - \gamma V(s_i')\right) \tag{7.6}$$

der gesamten Fehlerfunktion hängt von der Q-Funktion des aktuellen Zustandes und der Bewertungsfunktion des Folgezustandes ab. Um nun aber, im Gegensatz zur Minimierung

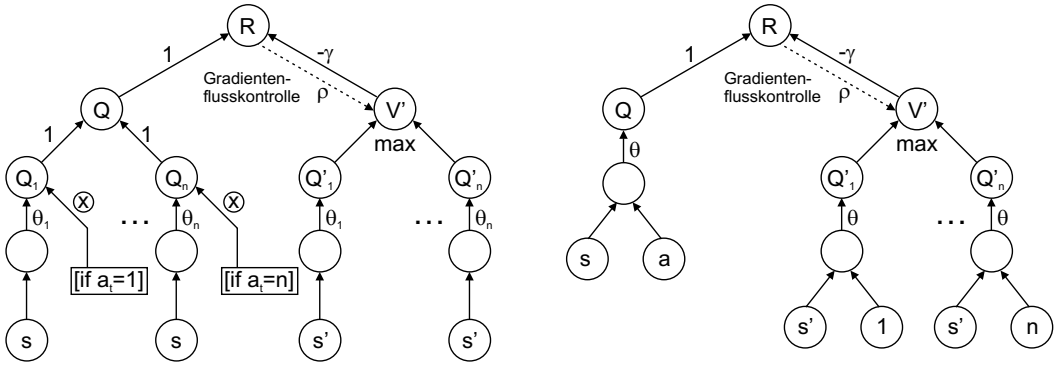


Abbildung 7.1: Beide Alternativen der vorgesehenen NRR-Architekturen (links: ein Netzwerk für jede Aktion, rechts: der monolithische Ansatz). Konnektoren mit gleichen Parametern sind mit Shared-Weights miteinander verknüpft. Im monolithischen Ansatz ist die Anzahl der Neuronen insgesamt größer, dafür ist jedoch eine Generalisierung über die Aktionen unmittelbar durch das die Q-Funktion repräsentierende Teilnetz möglich.

des Bellman-Residuums, den Fixpunkt der Bellman-Iteration anzustreben, der mit der TD-Lösung übereinstimmt, substituiert man

$$y_i \quad := \quad r_i + \gamma V(s'_i) \tag{7.7}$$

und minimiert iterativ

$$L \quad = \quad \sum_{i=1}^{l} F\left(Q(s_i, a_i) - y_i\right) \tag{7.8}$$

bis zur Konvergenz von Q (siehe Abschn. 2.5.2). Der Gradient ergibt sich in diesem Fall also zu

$$\frac{\partial L}{\partial \theta} \quad = \quad \sum_{i=1}^{l} F'(L_i) \frac{\partial}{\partial \theta} Q(s_i, a_i). \tag{7.9}$$

Offensichtlich unterscheiden sich die beiden Gradienten ausschließlich in der Richtung, nicht jedoch im Fehlerterm. Dies führt zu der Erkenntnis, dass beim Gradientenabstieg durch das Teilnetz für die Bewertungsfunktion durch Blockieren des Fehlerflusses vom Anstreben der Residual-Lösung auf das der TD-Lösung umgestellt werden kann. Aus diesem Grund wird im Back-Propagation-Algorithmus das Fehlersignal zwischen dem Reward-Cluster und dem Bewertungsfunktions-Cluster mit einem Skalar ρ multipliziert, der einen fließenden Übergang zwischen diesen beiden Zielfunktionen ermöglicht (Gradientenflusskontrolle). Für $\rho = 1$ erhält man das Bellman-Residuum, für $\rho = 0$ die TD-Lösung und $0 < \rho < 1$ führt zu entsprechenden Kompromisslösungen [6]. Optimalität ergibt sich im allgemeinen Fall also als die Lösung des Gleichungssystems

$$\Lambda \quad = \quad \sum_{i=1}^{l} F'\left(Q(s_i, a_i) - \gamma V(s'_i) - r_i\right) \frac{\partial}{\partial \theta} \left(Q(s_i, a_i) - \rho \gamma V(s'_i)\right) = 0. \tag{7.10}$$

Ein tiefer liegender Blick eröffnet die Ähnlichkeit der Funktionsweise der NRR mit dem TD-Learning. Denn mit $\rho = 0$, $F(x) = x^2$ und einer inkrementellen Lernregel ist NRR identisch mit dem Verfahren, falls es im Batch-Learning angewendet wird, also immer wieder die gleichen Daten betrachtet. Die NRR-Architektur erlaubt jedoch die Anwendung einer Reihe von Lernverfahren für neuronale Netze [70], die im TD-Learning nicht ohne Weiteres umgesetzt werden können.

7.1 Regularisierung in der Neural-Rewards-Regression

Ein Kernbaustein der Regularisierung besteht, wie ausführlich in Kap. 3 diskutiert wurde, im Berücksichtigen eines Regularisierungsterms Ω als Bestandteil der Fehlerfunktion, der von den Parametern des neuronalen Netzes abhängt. Das in Abschn. 3.4 eingeführte Weight-Decay ist ein Beispiel, für das Ω differenzierbar ist. Wir erhalten also die modifizierte Zielfunktion

$$L = \sum_{i=1}^{l} F(L_i) + \Omega(\theta) \tag{7.11}$$

$$= \sum_{i=1}^{l} F\left(Q(s_i, a_i) - \gamma V(s_i') - r_i\right) + \Omega(\theta), \tag{7.12}$$

und den dazugehörigen Gradienten

$$\Lambda = \sum_{i=1}^{l} F'\left(Q(s_i, a_i) - \gamma V(s_i') - r_i\right) \frac{\partial}{\partial \theta}\left(Q(s_i, a_i) - \rho\gamma V(s_i')\right) + \frac{\partial \Omega(\theta)}{\partial \theta}. \tag{7.13}$$

Der Regularisierungsterm ist nicht abhängig von der durch ρ festgelegten, angestrebten Optimalität.

Regularisierung kann aber sehr erfolgreich auch in anderer Form umgesetzt werden. In der Praxis bewähren sich stochastische Optimierungsverfahren, der Einsatz des Vario-Eta-Verfahrens und die Variation der Netzgröße. Zudem ist der Einsatz des Ensemble-Learnings erfolgversprechend (siehe Abschn. 3.4). Entscheidend ist die breite Anwendbarkeit dieser Technik. NRR kann mit einer Vielzahl von Lernverfahren und Regularisierungstechniken kombiniert werden, so dass eine problemangepasste und dateneffiziente Lösung des jeweiligen RL-Problems angestrebt wird.

7.2 Policy-Identifikation in der Neural-Rewards-Regression

Wie bei der Kernel-Rewards-Regression gibt es unterschiedliche Optionen zur Identifikation möglichst guter Policies. Wir diskutieren verschiedene Lernverfahren für die NRR und ihre Einordnung in den größeren Kontext.

7.2.1 Neural-Rewards-Regression als Fitted-Q-Iteration

Die NRR kann als Verallgemeinerung der Neural-Fitted-Q-Iteration aufgefasst, sie kann in der Tat gerade als NFQ angewendet werden. Dies geschieht durch ein wiederholtes Verknüpfen und Entkoppeln der Shared-Weights (siehe Alg. 7).

7.2.2 Verschränkte Iteration

Die NRR ist in dieser Form, wie auch die NFQ, an einen algorithmischen Rahmen gebunden, in dem wiederholt Regressionsprobleme gelöst werden. Dies kann durch einen integrierten Prozess abgelöst werden. Durch die Verknüpfung mit Shared-Weights werden die Parameter θ nach jedem Adaptionsschritt eines geeigneten Back-Propagation-Verfahrens instantan aktualisiert. Das auf diese Weise umgesetzte Verfahren bezeichnen wir als Neural-Rewards-Regression i.e.S., da es ihrer Motivation am Besten entspricht. Es handelt sich so um ein direktes Lösungsverfahren wie auch die EKRR (siehe Abschn. 6.6).

Algorithmus 7 Neural-Rewards-Regression als Fitted-Q-Iteration

Bedingungen: gegeben eine Menge von Transitionen $T = \{(s_1, a_1, r_1, s'_1), \ldots, (s_l, a_l, r_l, s'_l)\}$

Zusicherungen: bestimmt eine möglichst gute Policy

initialisiere Gewichte θ des NRR-Netzwerkes mit Shared-Weights zufällig, so dass das Netzwerk als $N(s,a) - \gamma \max_{a'} N(s',a')$ repräsentiert wird

solange die gewünschte Genauigkeit nicht erreicht ist **führe aus**

setze $t = t + 1$

löse die Gewichtsabhängigkeiten, so dass θ in θ_0 für $N(s,a)$ und θ_1 für $N(s',a')$ aufgeteilt ist

wende einen angemessenen Back-Propagation-Algorithmus zur Lösung des Regressionsproblems $\forall i : r_i = N(s_i, a_i) - \gamma \max_{a'} N(s'_i, a')$ bzgl. θ_0 an

verbinde die Gewichtsabhängigkeiten, so dass $\theta = \theta_0$

ende solange

setze $\pi(s) = \max_a N(s,a)$

gebe π zurück

7.2.3 Neural-Rewards-Regression als Verallgemeinerung der Neural-Fitted-Q-Iteration

Wir betrachten die NRR jedoch als Verallgemeinerung der NFQ und wollen dabei von den verschiedenen speziellen Ausprägungen der Parameter der Iterationstechniken profitieren. Dabei unterscheiden wir qualitativ zwischen instantaner und iterativer Adaption bzgl. Q-Funktion und Policy sowie zwischen den Batch- und Pattern-by-Pattern-Verfahren. Die NFQ ordnet sich dabei in die Klasse der iterativen Verfahren sowohl bzgl. Q-Funktion als auch Policy [83], Policy-Iteration und Temporal-Difference-Learning auf der Basis neuronaler Netze an anderer Stelle als spezielle Ausprägungen der NRR ein. Eine detaillierte tabellarische Darstellung ist in der Tab. 7.1 vorgenommen.

Tabelle 7.1: Kategorisierung der Iterationstechniken für die Neural-Rewards-Regression bzgl. instantaner und iterativer Adaption von Q-Funktion und Policy sowie bzgl. Batch- und Pattern-by-Pattern-Verfahren. Die Adaption erfolgt instantan, falls sich die adaptierte Funktion mit jedem Lernschritt verändert und damit Auswirkungen auf den weiteren Adaptionsprozess hat. Bei der iterativen Adaption erfolgt die Veränderung der Funktion erst nach Erreichen eines festgelegten Optimalitätkriteriums.

NRR	**Q-Funktion instantan**	**Q-Funktion iterativ**
Policy instantan		
Batch-Verfahren	Neural-Rewards-Regression i.e.S. mit Batch-Learning	entf.
Pattern-by-Pattern-Verfahren	Temporal-Difference-Learning mit neuronalen Netzen auf Batch-Daten	entf.
Policy iterativ		
Batch-Verfahren	Policy-Iteration mit neuronalen Netzen und Batch-Learning	Neural-Fitted-Q-Iteration mit Batch-Learning
Pattern-by-Pattern-Verfahren	Policy-Iteration mit neuronalen Netzen und Pattern-by-Pattern-Learning	Neural-Fitted-Q-Iteration mit Pattern-by-Pattern-Learning

Ausgehend von der NRR i.e.S. können nun auch verschiedene Kompromissansätze realisiert werden. TD-Learning etwa arbeitet inkrementell, während jede Abstufung hin zu einer größeren Batch-Size Varianten der NRR sind, die sich nicht mehr als TD-Learning qualifi-

zieren. Policy-Iteration mit neuronalen Netzen und NFQ verfügen über eine iterative und damit algorithmische Komponente, die gerade bei neuronalen Netzen als Approximatoren das Risiko der Policy-Degradation [29] erhöht und zu schwieriger zu kontrollierenden Lernverfahren führt. Abstufungen sind aber auch hier möglich, etwa durch die Frequenz der Aktualisierung der Gewichte im Teilnetz für die Folgezustände beim NFQ. Dies muss nicht zwangsläufig erst bei Konvergenz erfolgen und im Grenzfall der Aktualisierung nach jeder Iteration erreichen wir wieder die NRR i.e.S.

Policy-Iteration ist in die Kategorie Policy iterativ/Q-Funktion instantan einzusortieren, da erst die Q-Funktion die Bellman-Gleichung korrekt lösen muss, bevor die Policy aktualisiert wird. Dies wird iterativ unter Anwendung des Gradientenverfahrens erreicht.

Neben der hier vorgenommenen Kategorisierung besteht ein wesentlicher Unterschied der NRR im Allgemeinen zur NFQ, Policy-Iteration mit neuronalen Netzen und TD-Learning hinsichtlich des bereits diskutierten Aspektes der Regularisierung, der weiteren Gestaltungsspielraum zu Gunsten einer möglichst guten Generalisierung ermöglicht.

7.3 Verbesserung des Optimalitätskriteriums – Auxiliared-Neural-Rewards-Regression

Zur Verbesserung der Erwartungstreue erweitern wir die NRR so, dass das Auxiliared-Bellman-Residuum Gl. 4.47 (siehe Abschn. 4.2) minimiert wird.

In Abschn. 5.4 haben wir bereits gesehen, dass Erwartungstreue nicht vollständig erreicht werden kann oder im Falle von $H_Q = H_h$ sogar wieder auf die Fixpunkt-Lösung führt. Im Gegensatz zum Vorschlag in [3, 4] wählen wir H_h daher entweder als deutlich reichere Funktionsklasse als H_Q oder ausgestattet mit Vorwissen über den wahren Bellman-Operator, so dass $\hat{L}$ im Wesentlichen eine bessere Schätzung für $\|T'Q\|^2$ ist (siehe dazu Gl. 4.46).

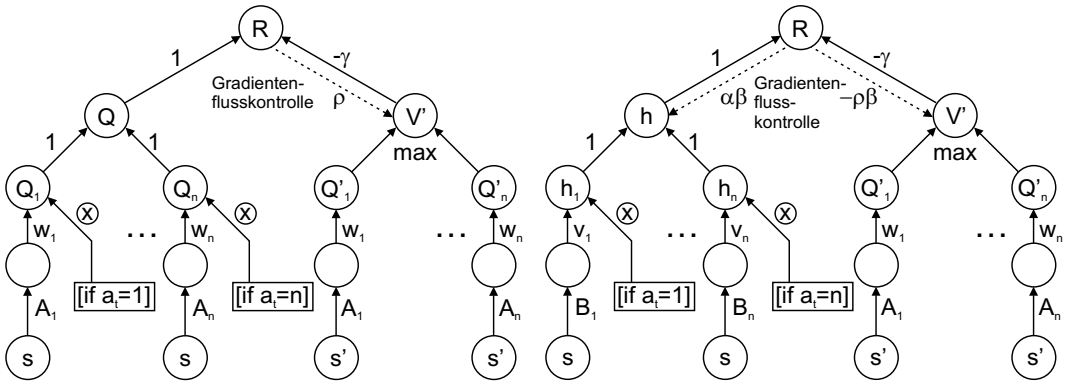


Abbildung 7.2: Die NRR-Architektur zur Berücksichtigung des Auxiliared-Bellman-Residuums, das zur Auxiliared-NRR führt. Beide Netzwerke werden simultan adaptiert. Der rechte Netzwerkteil wird verwendet, um TQ unabhängig von Q durch h zu approximieren. Damit wird der durch $T'Q$ ausgedrückte Fehlerterm in der Zielfunktion berücksichtigt.

Wenn θ die Parameter von Q und ω die von h sind, erhalten wir mit der Fehlerfunktion $F(x) = x^2$ demnach die Gradienten

$$\Delta\theta = \Lambda + \beta\rho\gamma \sum_{i=1}^{l} (h(s_i, a_i) - \gamma V(s'_i) - r_i) \frac{\partial}{\partial\theta} V(s'_i) \tag{7.14}$$

$$\Delta\omega = \alpha\beta \sum_{i=1}^{l} (h(s_i, a_i) - \gamma V(s'_i) - r_i) \frac{\partial}{\partial\omega} h(s_i, a_i), \tag{7.15}$$

wobei $0 \leq \beta \leq 1$ den Einfluss der Hilfsfunktion h und $\alpha \geq 1$ das Gewicht der Optimierung von h gegenüber Q steuert. Diese Komponenten sind wie folgt in die NRR-Architektur

eingebaut. Zusätzlich zum NRR-Netzwerk, das für die Minimierung von L zuständig ist, installieren wir ein zweites, ähnliches Hilfs-Netzwerk, bei dem die Q-Cluster durch entsprechende h-Cluster ersetzt werden. Dagegen bleibt der Netzwerk-Teil für V' gleich. Alle drei Q-Funktions-Netzwerkteile (für Q und V') bleiben weiterhin mit Shared-Weights verknüpft. Schließlich hat das Hilfsnetzwerk zwei Aufgaben zu erfüllen, das Maximieren von $\hat{L}$ bzgl. θ und das Minimieren von $\hat{L}$ bzgl. ω. Wegen des negativen Vorzeichens von $\hat{L}$ muss der Gradient durch V' auf der Seite des Hilfsnetzwerks in die entgegengesetzte Richtung gesendet werden. Die Parameter α müssen hinreichend groß gewählt werden, damit h den Bellman-Operator instantan approximieren kann.

7.4 Policy-Gradient-Neural-Rewards-Regression für kontinuierliche Aktionen

Die Kombination von NRR mit Policy-Gradient-Methoden führt in erster Linie zu einer bedeutenden Erweiterung der Problemklasse, die von der NRR erfasst wird. Denn es können kontinuierliche Aktionsräume betrachtet werden. Da die Berechnung von $V(s) = \max_a Q(s,a)$ nur in Spezialfällen, etwa bei Anwendung des Newton-Verfahrens, in polynomieller Zeit möglich ist, ermöglicht die direkte Darstellung der Policy durch eine Funktion, in diesem Fall durch ein neuronales Netz, die effiziente Anwendbarkeit kontinuierlicher Aktionsräume.

Obwohl als Alternativen etwa auch die Idee des Wire-Fitting [7] sowie einige kernelbasierte Szenarien (siehe Abschn. 6.7) vorliegen, ist der Policy-Gradient-Ansatz dagegen sehr viel allgemeiner und ermöglicht einen größeren Gestaltungsspielraum. Dazu ersetzen wir die Ensembles von Q-Funktions-Netzen für die diskreten Aktionen durch ein einzelnes Netzwerk zur Bestimmung von $Q(s,a)$ für kontinuierliche Zustände und Aktionen. Für den Folgezustand wird der Ausdruck $Q(s', \pi(s'))$ evaluiert. Dabei fassen wir $\pi : S \to A$ mit Parametern ψ als Funktion auf, die die optimale Policy anstrebt. Dazu muss sich $Q(s', \pi(s'))$ dem Wert von $\max_{a'} Q(s', a')$ annähern. Dies wird erreicht durch das Maximieren der Q-Funktion für die Folgezustände, simultan mit der Regression der Reward-Funktion.

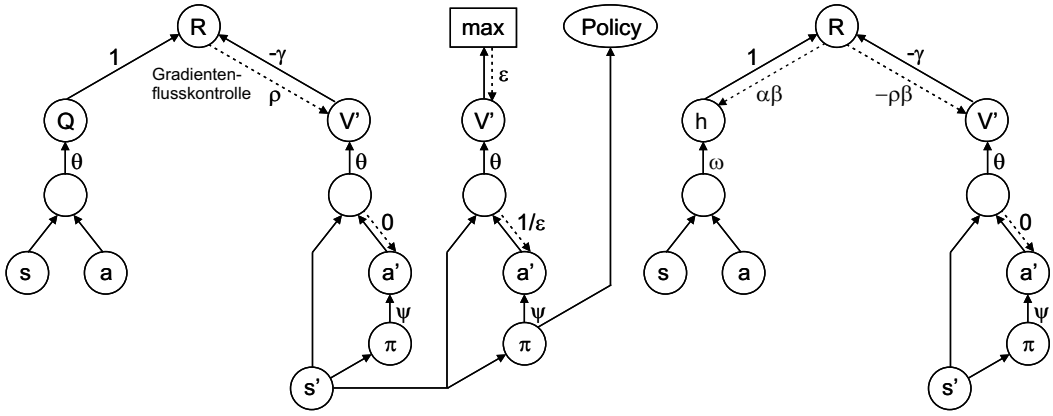


Abbildung 7.3: Die Policy-Gradient-NRR-Architektur. Die zusätzlichen Bausteine werden verwendet, um das Policy-Netzwerk zu trainieren, deren Aufgabe darin besteht, eine möglichst gute Policy abzubilden.

Auf diese Weise wird durch das Ausnutzen der funktionalen Zusammenhänge zwischen Q-Funktion und Policy eine Art Batch-On-Policy-Iteration oder Batch-Actor-Critic-Iteration realisiert. Die Gradientenflusskontrolltechnik in Verbindung mit Shared-Weights ist nach wie vor ausreichend für die Konstruktion einer entsprechenden Architektur. Im Netzwerk für den Folgezustand wird der Gradientenfluss durch das Policy-Netzwerk gesperrt, so dass die Policy die Regression an die Rewards nicht beeinflussen kann. Im erweiterten Netzwerk-Teil, das

in Abb. 7.3 skizziert und in dem die Bewertungsfunktion maximiert wird, wird der Gradientenfluss durch das Bewertungsfunktions-Teilnetz durch Multiplikation mit einem hinreichend kleinen ϵ soweit reduziert, dass nur die Policy zur Maximierung der Q-Funktion beiträgt. Die Richtung des Gradienten muss aber erhalten bleiben. Die gemeinsamen Gradienten ergeben sich dann zu

$$\begin{aligned}
\Delta\theta &= \sum_{i=1}^{l} \Big((Q(s_i,a_i) - \gamma Q(s'_i,\pi(s'_i)) - r_i) && (7.16)\\
&\quad \frac{\partial}{\partial\theta}\left(Q(s_i,a_i) - \rho\gamma Q(s'_i,\pi(s'_i))\right) + \frac{\varepsilon\partial}{\partial\theta} Q(s'_i,\pi(s'_i)) && (7.17)\\
&\quad +\beta\rho\gamma\left(h(s_i,a_i) - \gamma Q(s'_i,\pi(s'_i)) - r_i\right) && (7.18)\\
&\quad \frac{\gamma\partial}{\partial\theta} Q(s'_i,\pi(s'_i))\Big) + \frac{\partial\Omega(\theta)}{\partial\theta} && (7.19)\\
\Delta\omega &= \alpha\beta\sum_{i=1}^{l}\left(h(s_i,a_i) - \gamma Q(s'_i,\pi(s'_i)) - r_i\right)\frac{\partial}{\partial\omega} h(s_i,a_i) && (7.20)\\
\Delta\psi &= \sum_{i=1}^{l}\frac{d}{\varepsilon d\psi}\pi(s'_i)\frac{\varepsilon\partial}{\partial\pi(s'_i)} Q(s'_i,\pi(s'_i)) && (7.21)\\
&= \sum_{i=1}^{l}\frac{\partial}{\partial\psi} Q(s'_i,\pi(s'_i)), && (7.22)
\end{aligned}$$

wobei bzgl. θ und ω minimiert und bzgl. ψ maximiert wird. Eine möglichst gute Policy kann nach wie vor ausschließlich durch Anwendung von Back-Propagation mit Shared-Weights bestimmt werden. Nach Erreichen der Konvergenz wird die Policy durch das Netzwerk für π evaluiert, ohne dass dazu die Q-Funktion als Zwischenergebnis zur Hilfe genommen werden muss.

7.4.1 Steigerung der Generalisierungsfähigkeit bei paralleler Regularisierung von Q-Funktion und Policy

Wir gehen davon aus, dass mit der Erweiterung der Problemklasse auf kontinuierliche Aktionsräume und damit der Einführung eines expliziten Policy-Netzes die Dateneffizienz weiter gesteigert werden kann. Wir profitieren auf der einen Seite generell von den Vorteilen der Policy-Gradient-Verfahren, die eine Parametrisierung der Policy ermöglichen. Unabhängig davon sind die Policies i.d.R. auch einfacher darstellbar als die Q-Funktionen, dem kann bei der Wahl eines geeigneten Hypothesenraumes Rechnung getragen werden.

Auf der anderen Seite stellt sich aber auch die Frage, ob das kombinierte Lernen von Q-Funktion und Policy eine Verbesserung in Bezug auf den Bias-Variance-Trade-Off erzielt, ohne dass spezielles Vorwissen über die Gestalt der Policy einfließt. Dies kann erreicht werden, indem man über die Komplexität beider Hypothesenräume $H_Q \subset C_Q$ und $H_\pi \subset C_\pi$, die wiederum Teilmengen der jeweiligen Konzeptklassen sind, im Sinne der strukturellen Risiko-Minimierung eine gemeinsame optimale Wahl trifft.

Wir betrachten dazu eine Funktion $\Pi : 2^{C_Q} \to 2^{C_\pi}$, die von der Menge aller möglichen Q-Funktionsräume in die Menge aller möglichen Policy-Funktionsräume abbildet, so dass für $\Pi(H_Q) = H_\pi$ der Zusammenhang

$$\begin{aligned}
\forall Q \in H_Q : \exists \pi \in H_\pi \quad &: \quad \forall s \in S : \pi(s) = \arg\max_a Q(s,a)\\
\forall \pi \in H_\pi : \exists Q \in H_Q \quad &: \quad \forall s \in S : \pi(s) = \arg\max_a Q(s,a)
\end{aligned}$$

gilt. Das heißt, $\Pi(H_Q)$ enthält eine zu Q korrespondierende Policy für alle $Q \in H_Q$ und umgekehrt. Die Funktion Π existiert offensichtlich, da für RL-Verfahren, die keine explizite Darstellung der Policy benutzen, stets $H_\pi = \Pi(H_Q)$ ist.

Der Fragestellung, ob eine andere Wahl von H_π von Vorteil ist, wollen wir uns zuwenden und betrachten dazu eine einfache Cerebellar-Model-Articulation-Controller-Architektur (CMAC), bei der der Zustandsraum in Kacheln eingeteilt, also diskretisiert wird, als Regressor sowohl für die Q-Funktion als auch für die Policy [107, 110]. Diese Approximatoren werden auf das Wet-Chicken-Benchmark-Problem [111] angewendet, der in Abschn. 8.1.2 noch detailliert beschrieben wird. Mit ihnen verläuft der Vorgang des Lernens wesentlich effizienter als mit nicht-linearen neuronalen Netzen, die prinzipiellen Beobachtungen lassen sich auch mit diesen Regressoren nachvollziehen.

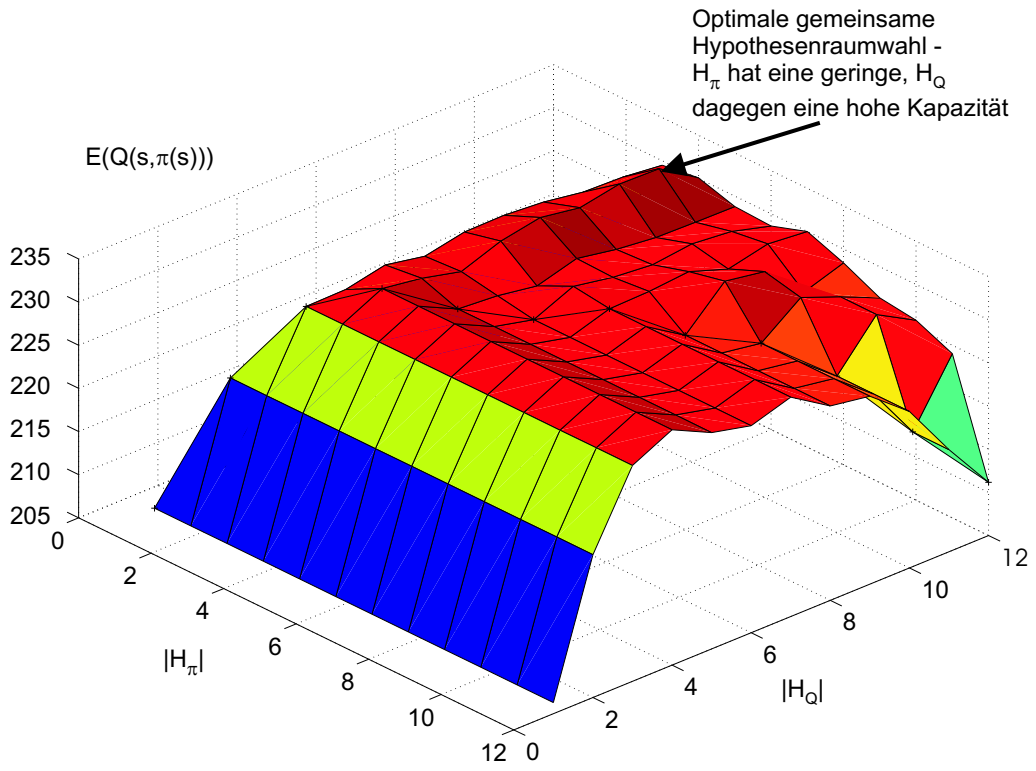


Abbildung 7.4: Verschränkter Bias-Variance-Trade-Off für Q-Funktion und Policy unter Anwendung von Tile-Coding-Approximatoren jeweils für Q-Funktion und Policy. Die Performance ist gemittelt über 100 Versuche. Entscheidend ist, dass die höchste Performance nicht an einem Punkt auf der Diagonalen mit $|H_Q| = |H_\pi|$ liegt, sondern für Kapazitäten erreicht wird, für die H_Q komplexer und H_π einfacher sind.

Dabei wird der Bias-Variance-Trade-Off über die Anzahl der Kacheln kontrolliert. Der Hypothesenraum $\Pi(H_Q)$ kann in diesem Fall einfach angegeben werden. Der verschränkte Trade-Off entsteht dann dadurch, dass die Anzahl der Kacheln für $\Pi(H_Q)$ und damit die Anzahl der Parameter reduziert sowie der Bias bzgl. H_π vergrößert wird, während die Anzahl der Kacheln für H_Q so gewählt werden kann, dass sie nicht zu H_π korrespondieren muss.

In Abb. 7.4 sind die Ergebnisse dieses Versuches dargestellt. Man sieht, was auch im Allgemeinen gilt, dass der optimale verschränkte Trade-Off, auf dem die maximale Performance erzielt wird, nicht auf der Diagonalen liegt, auf der H_Q und H_π zueinander korrespondieren, also $H_\pi = \Pi(H_Q)$. Am insgesamt optimalen Punkt ist die Komplexität von H_π geringer und die von H_Q größer als am optimalen Punkt auf der Diagonalen. Damit zeigt sich, dass tatsächlich eine Verbesserung der Generalisierung durch eine unabhängigere Betrachtung von Q-Funktion und Policy erreicht werden kann.

7.5 Recurrent-Neural-Rewards-Regression für partiell beobachtbare Systeme

Recurrent-Neural-Rewards-Regression (RNRR) ist eine Variante der NRR, die für die Problemklasse der POMDP entworfen ist. Die Grundidee der Kombination von NRR mit rekurrenten neuronalen Netzen besteht darin, die Markov-Eigenschaft im internen Zustand des RNN zu rekonstruieren. Während der Zustand s des Prozesses diese zwar nicht unbedingt erfüllt, kann dies aber für eine hinreichend große Historie von Zuständen der Fall sein. Dies macht sich die RNRR zu Nutze, indem sie mit Hilfe eines rekurrenten Netzes diese Informationen in den internen Zustand $\mathbf{x}$ aggregiert. Dabei wird, wie in Abschn. 2.2.3 argumentiert wird, von der Approximierbarkeit von POMDP durch MDP höherer Ordnung ausgegangen.

Aus einer anderen Perspektive betrachtet, besteht das Prinzip dieses Ansatzes in dem Ausnutzen der Informationen über die zu Grunde liegende Dynamik, die von einem RNN simuliert wird. Denn ist die Dynamik oder eine ihrer Approximationen bekannt, könnte diese auch als Grundlage für dynamische Programmierung benutzt werden.

In einer dritten Interpretation handelt es sich bei der Modellierung mittels RNN um einen speziellen Feature-Generator, der aus vergangenen Beobachtungen unter Nebenbedingungen, vor allem der Selbstähnlichkeit der zeitlichen Entwicklung, Features erzeugt, auf deren Grundlage die NRR arbeiten kann. Die Erweiterung der NRR um rekurrente Netze ist

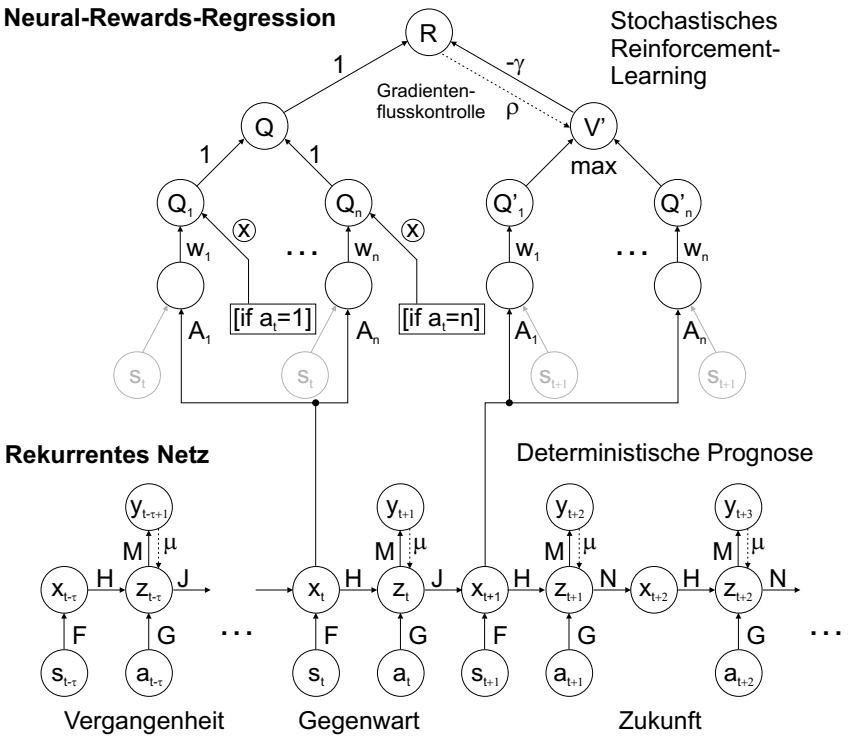


Abbildung 7.5: Die RNRR-Architektur. Das NRR-Teilnetz ist mit einem speziellen RNN verbunden, so dass MDP höherer Ordnung berücksichtigt werden können. Das RNN hat dabei die Aufgabe, die internen Zustände $\mathbf{x}_t$ mit der Markov-Eigenschaft auszustatten, damit das NRR auf Markov-Zuständen arbeiten kann.

für industrielle Anwendungen von entscheidender Bedeutung. Obwohl das zu regelnde System i.d.R. bereits viele Mess- und Stellgrößen zur Verfügung stellt, ist seine Komplexität so hoch, dass die beobachtbaren Größen das System kaum vollständig beschreiben können. Dieser Mangel wird durch die Rekonstruktion der Markov-Eigenschaft durch eine hinreichend große Historie von Zuständen beseitigt oder zumindest abgeschwächt.

7.5.1 Aufbau

Die RNRR-Architektur besteht nun aus einem NRR-Teilnetz und einem RNN, das damit verbunden ist. Das rekurrente Netz unterscheidet sich von denen in Abschn. 4.4 beschriebenen in zwei wesentlichen Punkten. Zum Einen werden zwei semantisch verschiedene Arten interner Zustände benötigt, die wir im Folgenden als dynamik-interne Zustände und als Markov-Zustände bezeichnen. Ein dynamik-interner Zustand entspricht dem internen Zustand klassischer RNN und soll die Informationen enthalten, die nötig sind, um die zukünftige Entwicklung vorhersagen zu können. Ein Markov-Zustand hingegen entspricht dem Zustand, der für das Reinforcement-Learning herangezogen wird. Zu einem festen Zeitpunkt t enthält der Markov-Zustand weniger Informationen als der dynamik-interne Zustand, da ihm die zum Zeitpunkt t ausgeführte Aktion noch nicht zugeführt wurde. Um diese Architektur zu gestalten, gehen wir von den Übergangsgleichungen

$$\mathbf{x}_t = \tanh(F\mathbf{s}_t + J\mathbf{z}_{t-1}) \tag{7.23}$$

$$\mathbf{z}_t = Ga_t + H\mathbf{x}_t \tag{7.24}$$

aus. Dabei ist $\mathbf{x}_t$ der Markov-Zustand und $\mathbf{z}_t$ der dynamik-interne Zustand. Die Vorhersage des Folgezustandes erfolgt dann mittels

$$M\mathbf{z}_t \rightarrow \mathbf{s}_{t+1}, \tag{7.25}$$

muss also vom dynamik-internen Zustand abhängen. Im Overshooting-Bereich erfolgt der Übergang von $\mathbf{z}$ nach $\mathbf{x}$ durch

$$\mathbf{x}_t = \tanh(N\mathbf{z}_{t-1}), t > i+1, \tag{7.26}$$

also mit einer anderen Matrix N. Dies ist der zweite wesentliche Unterschied und notwendig, um dynamische Konsistenz zu gewährleisten. Im Gegensatz zum Dynamical-Consistent-Neural-Network, in dem die Prognose als Eingabe für den Folgezustand verwendet wird, kann wegen der Stochastizität des Prozesses diese Annahme hier nicht gemacht werden.

Die Markov-Zustände $\mathbf{x}_t$ und $\mathbf{x}_{t+1}$ werden neben a_t nun als Eingabe für die NRR verwendet und entsprechen s_t und s_{t+1} bzw. s'_t. Das sind jeweils die Zustände, die zum jeweiligen Trainingsdatum gehören, also unmittelbar beim Übergang in den Overshooting-Teil. Kann die Markov-Eigenschaft tatsächlich rekonstruiert werden, ermittelt die NRR die Bewertungsfunktion für den entsprechenden Zustandsraum. Wie in Abschn. 4.4 ausgeführt wird, eignen sich RNN zur Approximation deterministischer dynamischer Systeme. Obwohl MDP und MDP höherer Ordnung im Allgemeinen stochastische Prozesse sind, stellt dies keine substanzielle Einschränkung dar. Der Markov-Zustand erfüllt bereits die Markov-Eigenschaft und damit die notwendigen Kriterien für die NRR, auch wenn die Vorhersage der Zukunft nicht fehlerfrei möglich ist. Wir werden im nächsten Abschnitt zeigen, dass gerade dieses Approximationsmodell geeignet ist, um die Rekonstruierbarkeit der Markov-Eigenschaft zu gewährleisten.

Der Informationsfluss zwischen RNN und NRR ist nicht statisch organisiert, also RNN und NRR müssen nicht getrennt oder nacheinander adaptiert werden, sondern die Gradienten vom Reward-Cluster können bis zu den Eingaben des RNN, also den externen Zuständen und Aktionen, fließen und, i.S. der oben genannten dritten Interpretation, zur Ausgestaltung eines optimalen Feature-Raumes beitragen, der sich neben dem Approximieren der Dynamik vor allem am Approximieren der Rewards orientiert.

7.5.2 Rekonstruierbarkeit von Markov-Zuständen mit rekurrenten neuronalen Netzen

Wir haben gesehen, dass rekurrente neuronale Netze deterministische dynamische Systeme beliebig genau approximieren können [91]. Verwendet man den quadratischen Fehler als Fehlermaß für die Prognose, sind RNN in der Lage, die zukünftigen Zustände im Erwartungswert

vorherzusagen, selbst wenn der zu Grunde liegende Prozess stochastisch ist. Das folgende Theorem zeigt, dass es in diesem Fall möglich ist, die Markov-Eigenschaft zu rekonstruieren, die internen Zustände des RNN also Markov-Zustände sind, was die Anwendbarkeit der RNRR untermauert.

Satz 6. *[101] Sei $M = (S, P)$ ein ergodischer Markov-Prozess mit abzählbar vielen Zuständen $s \in S$ der Ordnung $\tau + 1$ und P die entsprechende Übergangsmatrix. Sei ferner* RNN *ein rekurrentes neuronales Netz mit zeitlicher Entfaltung der Stufe τ und Overshooting-Länge ν. Dann ist $M' = (X, P')$ mit X dem internen Zustandsraum von* RNN *und $P' = P'(P, S, X)$ übertragen auf X, ein zu M identischer Markov-Prozess erster Ordnung, also $P\left(s^i_{j+1} \middle| \mathbf{x}^i_j\right) = P\left(s^i_{j+1} \middle| \left(s^i_j, \ldots, s^i_{j-\tau}\right)\right)$, falls $\nu \to \infty$ und* RNN *in der Lage ist, die erwarteten zukünftigen Zustände für alle $s \in S$ beliebig genau vorherzusagen.*

Beweis: Seien $\mathbf{w}^i_j = 1, i = j$ und $\mathbf{w}^i_j = 0, i \neq j$, mit $i, j \in \{1, \ldots, n\}$ und $n \in \mathbb{N}$ oder $n \to \infty$, die kanonischen Vektoren, die alle möglichen Zustände des Markov-Prozesses M beschreiben und die mit i nummeriert werden. Dann ist $(\mathbf{w}^i)^T P$ gerade der Vektor, der die Wahrscheinlichkeit der Folgezustände der entsprechenden Zustände beschreibt. Daher gilt

$$(\mathbf{w}^i)^T P^j \mathbf{e} = \mathbf{v}^j_i \tag{7.27}$$

$$\text{mit } \mathbf{e}_k = k \tag{7.28}$$

mit $\mathbf{v}^j_i$ der Nummer des $j$ten erwarteten Folgezustandes des Zustandes mit der Nummer i, da $\mathbf{e}$ über alle möglichen Folgezustände integriert. Wenn wir nun über alle möglichen Vorgängerzustände verallgemeinern, erhalten wir

$$\mathbf{I} P^j \mathbf{e} = P^j \mathbf{e} \tag{7.29}$$

$$= \mathbf{v}^j. \tag{7.30}$$

mit $\mathbf{v}^j$ dem Vektor der Wahrscheinlichkeiten aller möglichen $j$ten Folgezustände. Durch Substitution erhalten wir

$$P \mathbf{v}^j = \mathbf{v}^{j+1}. \tag{7.31}$$

Auf Grund der Ergodizität von M führt die sukzessive Anwendung von $(\mathbf{w}^i)^T P$ für ein beliebiges i zur Konvergenz gegen die stationäre Verteilung des Markov-Prozesses M. Daher muss die sukzessive Anwendung von P den vollständigen Raum $\mathbb{R}^n$ aufspannen. Also ist $(\mathbf{e}, P\mathbf{e}, \ldots, P^{\nu-1}\mathbf{e})$, mit $\nu \to \infty$ eine Matrix mit vollem Rang. Das lineare Gleichungssystem

$$P(\mathbf{e}, \mathbf{v}^1, \ldots, \mathbf{v}^{\nu-1}) = (\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^\nu). \tag{7.32}$$

determiniert daher P vollständig und wir können

$$P = (\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^\nu) C, \tag{7.33}$$

schreiben, wobei C existiert. Angenommen, es gelte nun das Gegenteil der Behauptung und mindestens ein $\mathbf{x}_m$ repräsentiert mindestens zwei verschiedene Markov-Zustände, o.B.d.A. 1 und 2. Dann wird RNN jedoch für beide Markov-Zustände die gleiche Prognose bieten und es folgt unmittelbar

$$\forall i : P_{1,i} = P_{2,i}, \tag{7.34}$$

so dass 1 und 2 die gleichen Markov-Zustände sind, was zu einem Widerspruch führt, der das Theorem beweist.

□

Bemerkung: Das Theorem kann für Markov-Entscheidungsprozesse verallgemeinert werden.

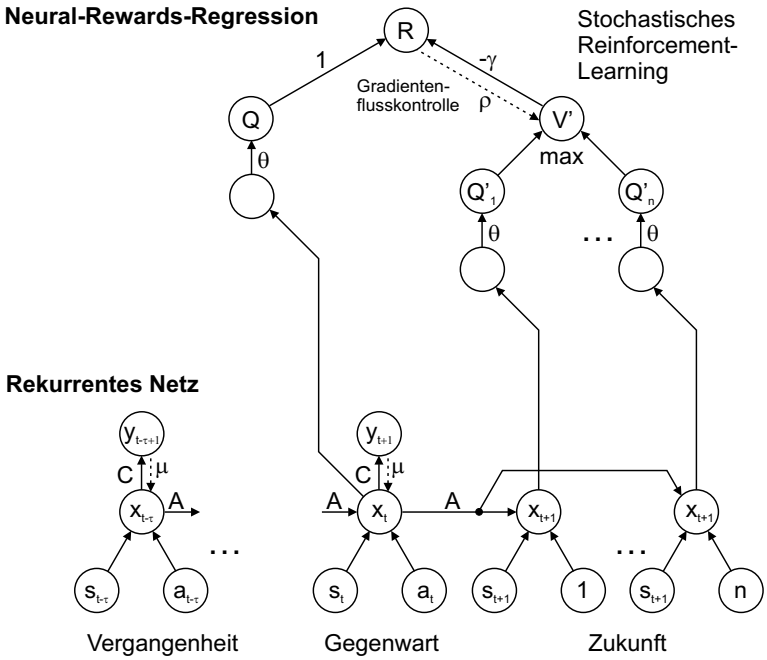


Abbildung 7.6: Die Normal-RNRR-Architektur. Das monolithische NRR ist verbunden mit einem klassischen RNN. Ein Overshooting-Teilnetz kann hier nicht berücksichtigt werden, da für den Zeitschritt $t+1$ bereits mehrere alternative Realitäten modelliert werden.

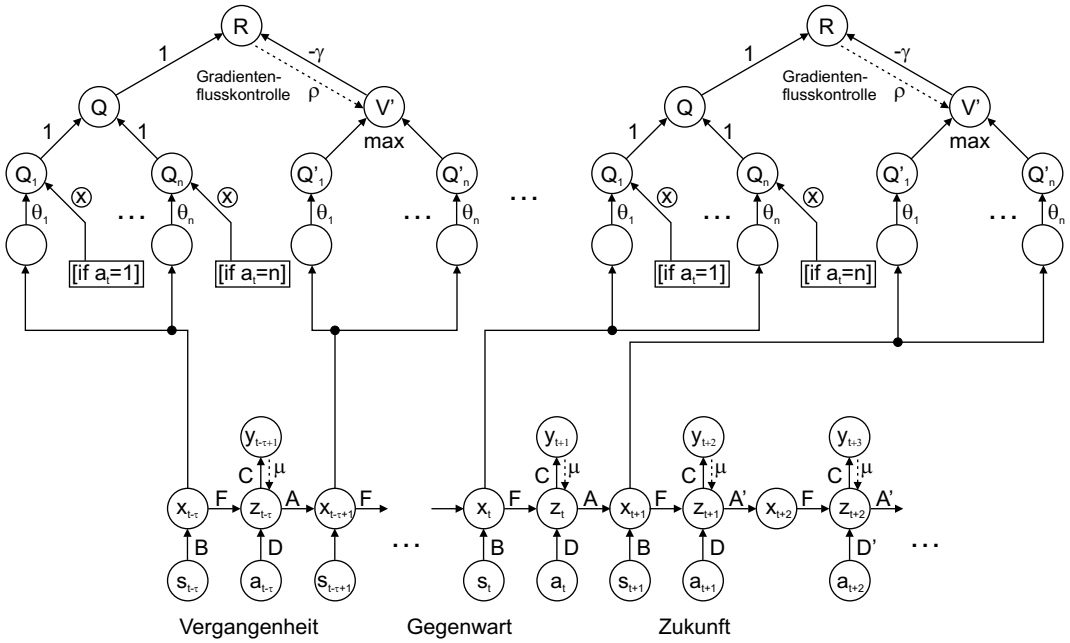


Abbildung 7.7: Die Multi-RNRR-Architektur. Hier sind mehrere NRR-Teilnetze mit dem RNN an den verschiedenen Zeitschritten verbunden. Für jede Stufe der zeitlichen Entfaltung wird also ein NRR-Teilnetz verwendet.

7.5.3 Weitere Varianten und spezielle Regularisierungstechniken

Eine einfache Modifikation besteht in dem Ersetzen der Matrix N durch J. Dies wird durch RNN motiviert, die nicht dynamisch konsistent sind, aber trotzdem erfolgreich in praktischen Anwendungen zum Einsatz kommen. Trotz des Fehlens der Eingabe von außen, was gerade das Fehlen von Datenkonsistenz verursacht, ist durch die Halbierung der Gewichte für die Übergangsmatrizen eine effektivere Regularisierung möglich, was für die Generalisierungsleistung des Gesamtsystems von Vorteil sein kann.

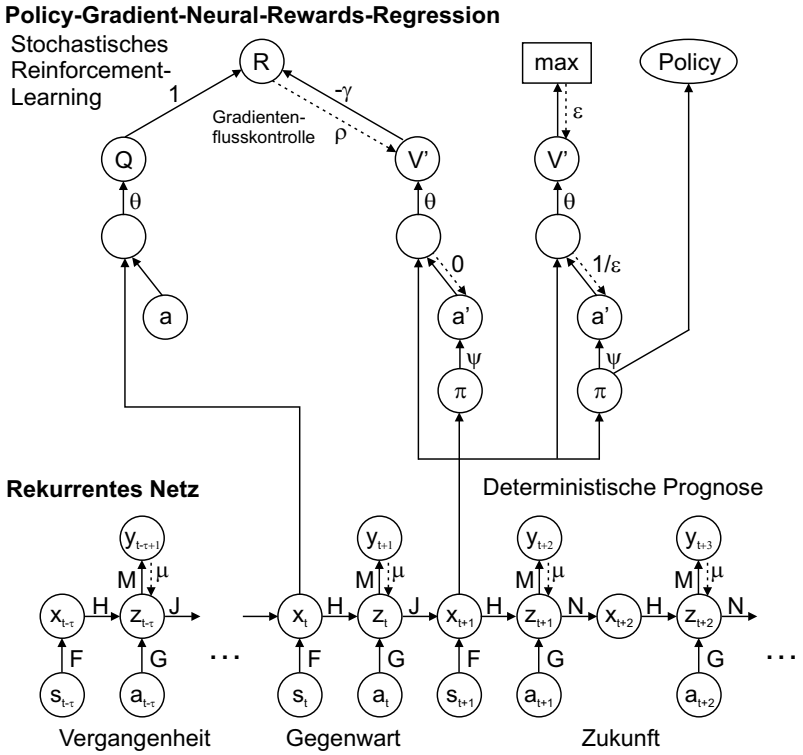


Abbildung 7.8: Die RPGNRR-Architektur als Kombination von PGNRR und RNRR.

7.5.3.1 Normal-Recurrent-Neural-Rewards-Regression

Eine ebenso wesentliche Modifikation besteht in dem Zurückgreifen auf klassische RNN. Dabei werden die Eingaben s_t und a_t zu einem Eingabevektor verschmolzen und keine Trennung in dynamik-interne und Markov-Zustände vollzogen. Dieses Architekturmodell kann nur ohne Overshooting funktionieren, da die Trennung der Aktionen für die Folgezustände durch eine Aufteilung in mehrere mögliche interne Folgezustände erfolgen muss. Zum Zeitpunkt $t+1$, also dem jeweiligen Trainingsdatum zugeordneten Folgezustand, werden dem RNN dann alle möglichen Aktionen in Kombination mit dem entsprechenden Folgezustand angeboten und $|A|$ unterschiedliche interne Folgezustände konstruiert, die dann als alternative Eingaben für die NRR dienen.

7.5.3.2 Modifikation der Gradientenflusskontrolle

Ferner besteht auch die Möglichkeit, den Gradientenfluss von der NRR durch das RNN zu sperren. Dann kann das RNN das dynamische System adaptieren, ohne dabei von der Maßgabe der Reward-Approximation beeinflusst zu werden. Hinsichtlich der Regularisierungsmöglichkeiten haben beide Ansätze ihre Vor- und Nachteile. Natürlich verzerrt der Einfluss der Rewards die Approximierbarkeit der Dynamik und damit die Erfüllbarkeit der

Markov-Eigenschaft. Andererseits ermöglicht der Gradientenfluss vom NRR durch das RNN die Features in optimaler Weise für das tatsächliche Problem, die Rewards zu approximieren, aufzubauen, wodurch die Anzahl der Gewichte reduziert und die Generalisierungsleistung erhöht werden könnte.

7.5.3.3 Multi-Recurrent-Neural-Rewards-Regression

Schließlich besteht darüber hinaus die Möglichkeit, die zeitliche Entfaltung in ihrer Symmetrie auch auf die NRR zu übertragen und dabei von der Generalisierungsleistung der RNN durch diesen Effekt zu profitieren. Dabei werden nicht nur die Tripel $(\mathbf{x}_t, a_t, \mathbf{x}_{t+1})$ als Eingabe für die NRR verwendet, sondern auch die jeder zeitlichen Verschiebung bis zum Grad der zeitlichen Entfaltung.

7.5.4 Recurrent-Policy-Gradient-Neural-Rewards-Regression

Die Kombination von PGNRR und RNRR führt zur Recurrent-Policy-Gradient-Neural-Rewards-Regression (RPGNRR) und stellt das bzgl. der zu bewältigenden Problemklasse mächtigste verfügbare vom NRR abgeleitete Verfahren dar. Das Prinzip der Kombination von NRR mit RNN bleibt bestehen, jedoch wird zusätzlich eine Verbindung vom Markov-Zustand zur Policy realisiert. Bleibt der Gradientenfluss auch an dieser Stelle bestehen, wird das rekurrente Netz bzgl. dreier Kriterien optimiert, der Approximation des dynamischen Systems, der Approximation der Rewards und dem Maximieren der Q-Funktion bzgl. der Policy.

Kapitel 8

Benchmarks und industrielle Anwendungen

Für den empirischen Nachweis sowohl der Informationseffizienzsteigerung als auch der Beherrschbarkeit breiter Problemklassen durch die Rewards-Regression untersuchen und erproben wir ihr Verhalten an Benchmarks und realen Regelungsproblemen. Wir konzentrieren uns dabei auf die Neural-Rewards-Regression und ihre Erweiterungen sowie die Kernel-Rewards-Regression in der Kombination mit Ridge-Regression.

8.1 Klassische Benchmarks

Zunächst betrachten wir klassische Benchmarks, die als bekannte und gut untersuchte Probleme gut geeignet sind, die Effektivität unserer Methoden nachzuweisen und einen Vergleich zu anderen Methoden zu ermöglichen. Das Ziel besteht hier darin, zu zeigen, dass die Informationseffizienz gesteigert werden kann, also bei besonders wenigen Beobachtungen, die gegebenenfalls noch durch eine zufällige Exploration gewonnen wurden, gute Policies zu bestimmen. Dabei spielt die Komplexität der Benchmarks eine untergeordnete Rolle.

8.1.1 Das Cart-Pole-Problem

Das Ziel beim Cart-Pole-Problem besteht darin, einen Stab (Pole) auf einem Wagen (Cart) möglichst lange zu balancieren und dabei gegebenenfalls mit dem Wagen einen zulässigen Bereich nicht zu verlassen. Der Zustandsraum ist, je nach Szenario, zwei- oder vierdimensional und besteht aus den physikalischen Größen x, $\dot{x}$, θ und $\dot{\theta}$, die Position und Geschwindigkeit

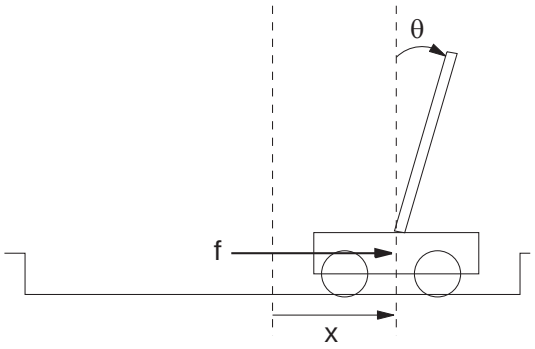


Abbildung 8.1: Illustration des Cart-Pole-Problems. Der Wagen (Cart) muss so bewegt werden, dass der Stab (Pole) nicht umkippt.

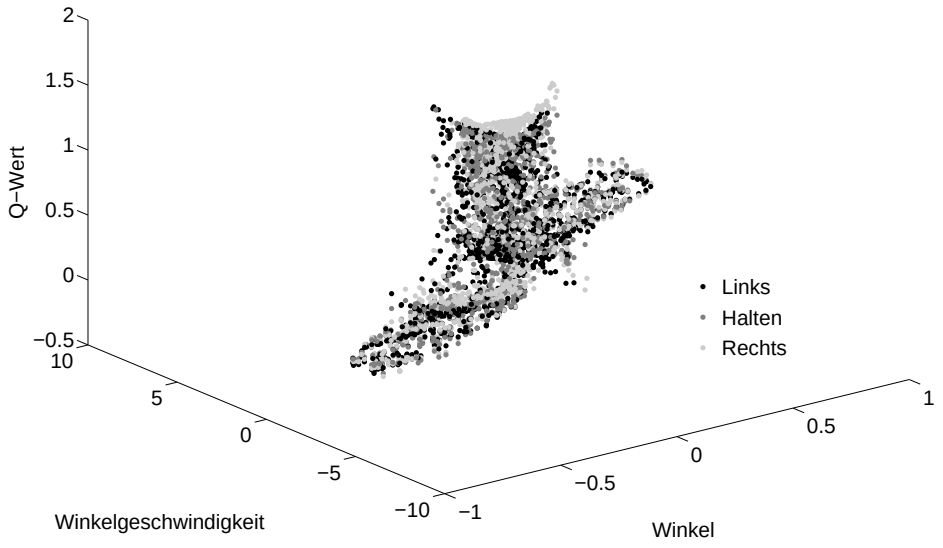


Abbildung 8.2: Exemplarische Q-Funktion für das Cart-Pole-Problem unter Anwendung der Kernel-Rewards-Regression auf 1500 Beobachtungen. Hier wird lediglich die Abhängigkeit von Winkel und Winkelgeschwindigkeit betrachtet. Die höchsten Q-Werte sind um den Null-Punkt zu beobachten, da der Stab sich dort in der Nähe seiner Ziellage befindet.

des Wagens sowie Winkel und Winkelgeschwindigkeit des Stabes bezeichnen. Der zweidimensionale Zustandsraum besteht nur aus den Größen θ und $\dot{\theta}$. In diesem Fall entfällt natürlich das zweite Ziel des Regelungsproblems. Der Aktionsraum ist diskret mit $A = \{-1, 0, 1\}$ und repräsentiert die Kraft, die auf den Wagen ausgeübt wird, um ihn in eine der beiden Richtungen zu beschleunigen. Um den Schwierigkeitsgrad zu erhöhen, verfügt das Problem über eine stochastische Komponente. Die Transitionen folgen den Regeln

$$f = af_m + 2\left(\text{rand} - \frac{1}{2}\right) f_r \tag{8.1}$$

$$c = \frac{f + p_{ml}\dot{\theta}^2 \sin(\theta)}{m_t} \tag{8.2}$$

$$\ddot{\theta} = \frac{g\sin(\theta) - \cos(\theta)c}{\frac{l}{2}\left(\frac{4}{3} - \frac{m_p \cos(\theta)^2}{m_t}\right)} \tag{8.3}$$

$$\ddot{x} = c - p_{ml}\ddot{\theta}\frac{\cos(\theta)}{m_t}. \tag{8.4}$$

Die Entwicklung der Dynamik wird numerisch durch Euler-Integration [37] gemäß

$$x_{\text{neu}} = x + \tau\dot{x} \tag{8.5}$$

$$\dot{x}_{\text{neu}} = \dot{x} + \tau\ddot{x} \tag{8.6}$$

$$\theta_{\text{neu}} = \theta + \tau\dot{\theta} \tag{8.7}$$

$$\dot{\theta}_{\text{neu}} = \dot{\theta} + \tau\ddot{\theta} \tag{8.8}$$

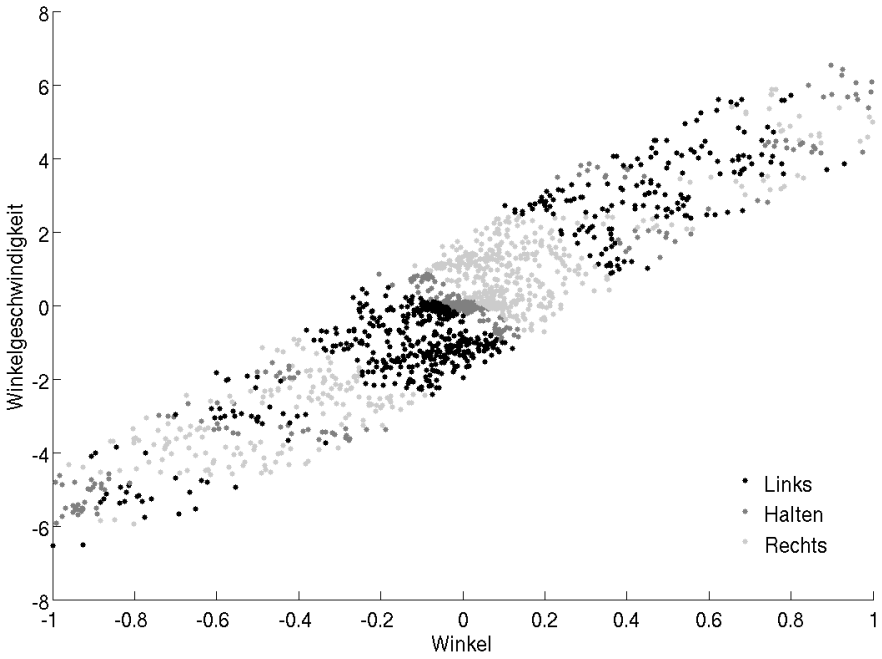


Abbildung 8.3: Exemplarische Policy für das Cart-Pole-Problem unter Anwendung der Kernel-Rewards-Regression auf 1500 Beobachtungen. Die Schwierigkeit besteht in der genauen Beschreibung der Grenzübergänge von den Aktionen Links nach Halten und Halten nach Rechts. In den Außenbereichen, in denen das Umfallen des Stabes nicht mehr verhindert werden kann, spielt die Auswahl der Aktion also keine Rolle mehr. Aus diesem Grund ist die Policy in diesem Bereich indifferent.

berechnet. Dabei ist rand eine Zufallszahl zwischen 0 und 1, f_m die auszuübende Kraft, f_r eine zufällige Kraftkomponente, $p_{ml} = \frac{1}{2}ml$ die Hälfte des Produktes aus Masse und Länge des Stabes, m_p die Masse des Stabes, $m_t = m_p + m_c$ die Summe aus der Masse des Wagens und der des Stabes sowie g die Fallbeschleunigung und τ die Dauer eines Zeitschritts. Die Reward-Funktion ist im zweidimensionalen Szenario

$$R(s,a,s') = \begin{cases} 0 & : \quad |\theta| < \theta_{\max} \\ -1 & : \quad \text{sonst} \end{cases} \tag{8.9}$$

mit $\theta_{\max} = \frac{\pi}{2}$ und im vierdimensionalen Fall, in dem Position und Geschwindigkeit des Wagens explizit berücksichtigt werden, entsprechend

$$R(s,a,s') = \begin{cases} 0 & : \quad |x| < x_{\max} \ \wedge \ |\theta| < \theta_{\max} \\ -1 & : \quad \text{sonst} \end{cases} \tag{8.10}$$

mit $\theta_{\max} = \frac{\pi}{30}$ und $x_{\max} = 2.4$. Im zweidimensionalen Szenario, auf das wir uns im Folgenden beziehen, werden die Werte gemäß

$$g = 9.8 \ \frac{\text{m}}{\text{s}^2} \tag{8.11}$$

$$m_c = 8.0 \text{ kg} \tag{8.12}$$

$$m_p = 2.0 \text{ kg} \tag{8.13}$$

$$l = 0.5 \text{ m} \tag{8.14}$$

$$f_m = 50.0 \text{ N} \tag{8.15}$$

$$f_r = 10.0 \text{ N} \tag{8.16}$$
$$\tau = 0.1 \text{ s} \tag{8.17}$$

gewählt. Man beachte dabei, dass der Zustand $s = \left(x, \dot{x}, \theta, \dot{\theta}\right)$ bzw. $s = \left(\theta, \dot{\theta}\right)$ aus den beobachteten Größen besteht.

8.1.1.1 Klassische Regelung

Das Cart-Pole-Problem ist ein klassisches Regelungsproblem und in der Literatur auch unter dem Begriff invertiertes Pendel bekannt. Um dieses oder ähnliche Probleme zu lösen, bedient man sich in der Regelungstechnik klassischer Verfahren, die auf Grund der Tatsache, dass das Modell analytisch (wie oben beschrieben) bekannt ist, nahezu perfekt funktionieren.

Im Falle des (zweidimensionalen) Cart-Pole-Problems ist die an den Sollwert heranzuführende Messgröße der Winkel θ des Stabes und die zu beeinflussende Stellgröße die auf den Wagen auszuübende Kraft f. Ein einfacher PID-Regler ist bereits in der Lage, dieses Problem zu beherrschen. Dies ist sowohl in seiner linearisierten Variante als auch im Original-Problem nachweisbar. In Abb. 8.4 sind drei Beispielkurven für die Anwendung von

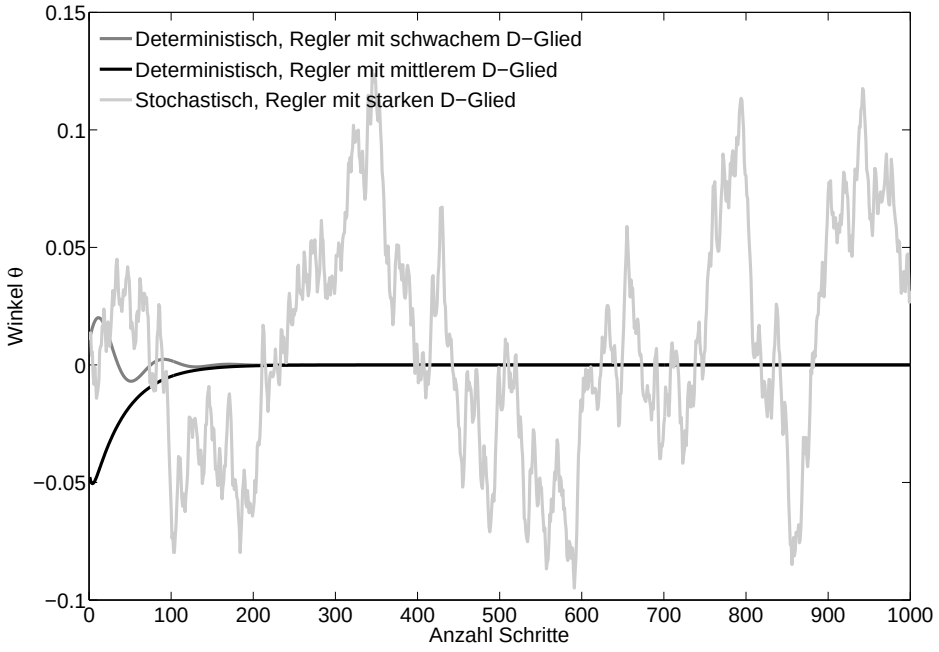


Abbildung 8.4: PID-Regelung für das Cart-Pole-Problem. Die Grafik zeigt den Winkel des Stabes im Verlauf der Zeit für die deterministische ($f_r = 0$ N) und die stochastische Variante ($f_r = 10$ N) des oben beschriebenen Cart-Pole-Problems. Dadurch, dass die Stochastizität die Wirkung der Aktion selbst verändert, entsteht der volatile Kurvenverlauf für letztere Einstellung.

PID-Reglern auf dem Cart-Pole-Problem gezeigt. In allen drei Fällen wurde $K_P = 160$ N gewählt. Die Einstellungen für das Differenzialglied sind gemäß $K_D = 50$ Ns (schwach), $K_D = 250$ Ns (mittel) und $K_D = 500$ Ns (stark) sowie das Integralglied in allen drei Fällen auf $K_I = 0$ Ns^2 gesetzt. Ein ausgeprägteres Differenzialglied verlangsamt die Einschwingzeit, also die Konvergenz, aber reduziert auch den Effekt von Überschwingern. Diese Zusammenhänge gelten nicht nur für diesen Benchmark, sondern auch im allgemeinen Fall.

Zum Vergleich ist der Verlauf des Winkels des Stabes auch für die Anwendung der KRR gezeigt (siehe Abb. 8.5). Durch die Einschränkung auf diskrete Aktionen ist der Kurven-

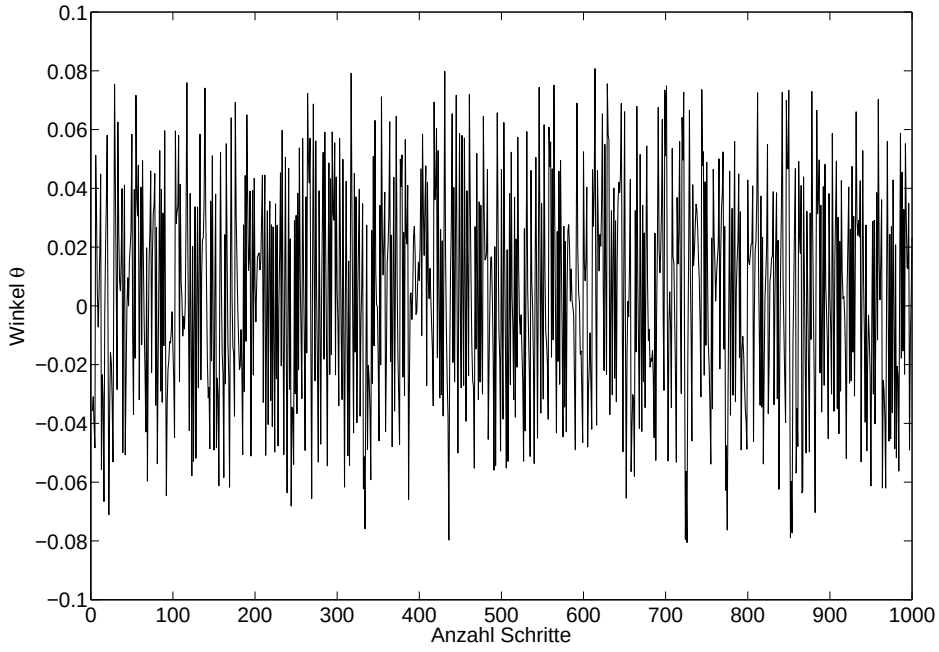


Abbildung 8.5: Kernel-Rewards-Regression für das Cart-Pole-Problem. Die Grafik zeigt den Winkel des Stabes im Verlauf der Zeit für die stochastische Variante ($f_r = 10$ N) des oben beschriebenen Cart-Pole-Problems. Im Vergleich zum PID-Regler (siehe Abb. 8.4) streut der Winkel stärker. Das hängt damit zusammen, dass die Aktionen nur diskret gewählt werden können, während der PID-Regler auf kontinuierliche Aktionen zurückgreifen und damit einen stabileren Kurvenverlauf erzeugen kann.

verlauf wesentlich dynamischer, die Auslenkungen müssen stärker kompensiert werden. Das Verfahren ist ebenfalls erfolgreich, obwohl kein Modell, sondern stattdessen 1500 Beobachtungen zur Verfügung standen.

8.1.2 Das Wet-Chicken-Problem

Der Wet-Chicken-Benchmark [111] ist ein stochastisches Regelungsproblem mit eindimensionalem Zustandsraum und diskretem Aktionsraum. Dabei sind $S = [0, \ldots, s_{\max}]$ und $A = \{-1, 0, 1\}$. Der Zustand entspricht der Position eines Kanus auf einem Fluss. Fährt der Kanute bis zur Position $s_{\max}$, dann ist ein Wasserfall erreicht, der das Kanu zum Absturz führt. In diesem Fall muss der Kanute erneut bei $s = 0$ beginnen. Der Reward ist proportional zur Position des Kanus und entspricht dem Applaus eines Publikums. Die Aktionen lassen sich mit den Begriffen Zurückrudern, Halten und Treiben bezeichnen. Im Einzelnen ist

$$P(s'|s,a) \quad = \quad \begin{cases} \frac{1}{v_s} & : \quad m_{\min} < s' < m_{\max} \\ \frac{\left[\frac{v_s}{2} - h(s+a)\right]_+ + \left[h(s+a) + \frac{v_s}{2} - s_{\max}\right]_+}{v_s} & : \quad s' = 0 \\ 0 & : \quad \text{sonst} \end{cases} \tag{8.18}$$

mit

$$h(x) \quad = \quad \begin{cases} 0 & : \quad x < 0 \\ s_{\max} & : \quad x > s_{\max} \\ x & : \quad \text{sonst} \end{cases} \tag{8.19}$$

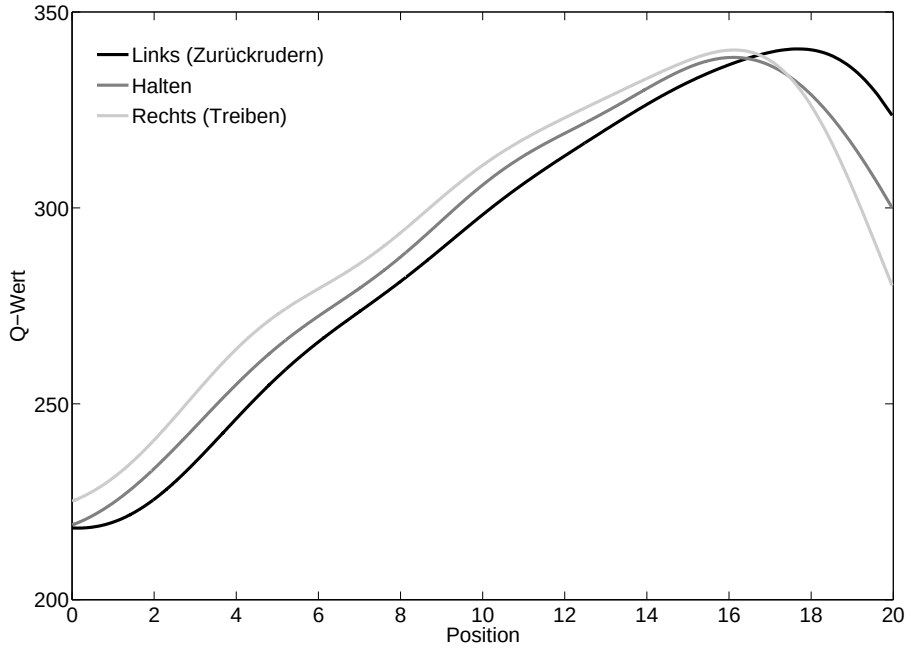


Abbildung 8.6: Exemplarische Q-Funktionen für das Wet-Chicken-Problem unter Anwendung der Kernel-Rewards-Regression auf 1500 Beobachtungen. Der Wasserfall befindet sich an Position 20. Die Schwierigkeit bei diesem Problem besteht in der indirekten Bestimmung des Schnittpunktes der Q-Funktionen für die Aktionen Links (Zurückrudern) und Rechts (Treiben).

$$m_{\min} = \max\left(s + a - \frac{v_s}{2}, 0\right) \tag{8.20}$$

$$m_{\max} = \min\left(s + a + \frac{v_s}{2}, s_{\max}\right) \tag{8.21}$$

$$[x]_+ = \begin{cases} x & : \quad x > 0 \\ 0 & : \quad \text{sonst} \end{cases}. \tag{8.22}$$

Auf Grund der Stochastizität, deren Grad durch v_s bestimmt wird, besteht die optimale Policy nicht darin, sich unmittelbar vor den Wasserfall zu positionieren, sondern einen ausreichenden Abstand zu halten. Die Parameter werden mit $s_{\max} = 20$ und $v_s = 4$ gesetzt. Die Reward-Funktion ist definiert als

$$R(s, a, s') = s. \tag{8.23}$$

In einer besonderen Variante, die später als partiell beobachtbares Problem herangezogen wird, unterliegt die Stärke der Strömung f selbst einer Stochastizität, die Dynamik wird gemäß

$$P(s'|(s,f),a) = \begin{cases} 1 & : \quad (s' = 0 \wedge s + a + f \geq s_{\max}) \\ & : \quad \vee (s' = \max(s + a + f, 0) \wedge s + a + f < s_{\max}) \\ 0 & : \quad \text{sonst} \end{cases} \tag{8.24}$$

$$P(f'|(s,f),a) = \begin{cases} \frac{1}{v_f} & : \quad \max\left(f_{\min}, f - \frac{v_f}{2}\right) < f' < \min\left(f_{\max}, f + \frac{v_f}{2}\right) \\ \frac{\left[\frac{v_f}{2} - (f - f_{\min})\right]_+}{v_f} & : \quad f' = f_{\min} \\ \frac{\left[f - f_{\max} + \frac{v_f}{2}\right]_+}{v_f} & : \quad \text{sonst} \end{cases} \tag{8.25}$$

verändert, wobei $v_f = 0.1$ und $f_{\min} = -1$ sowie $f_{\max} = 1$ gesetzt werden.

8.1.3 Ergebnisse

8.1.3.1 Cart-Pole-Problem

Wir betrachten zunächst das Cart-Pole-Problem, das in der oben beschriebenen Variante auch in [57] untersucht wird. Als Vergleich ziehen wir außerdem die in [83] beschriebenen Ergebnisse heran, die als Nachweis für eine Dateneffizienzsteigerung durch die NFQ präsentiert wurden. In den Tabn. 8.1, 8.2 und 8.3 sind entsprechende Ergebnisse für die NRR und die KRR aufgeführt.

Tabelle 8.1: Anzahl der erfolgreichen Lernversuche (von insgesamt je 50) für kurze Evaluationsläufe für das Cart-Pole-Problem. Ein Versuch wird als erfolgreich gewertet, falls der Stab mindestens 3000 Schritte balanciert werden kann. Zum Vergleich sind Ergebnisse aus [83] eingetragen. Weight-Decay bedeutet, dass die Zielfunktion um einen Weight-Decay-Term erweitert wird, 2× # Lernschritte heißt, dass doppelt so lange trainiert wurde, $x \times y$ Neuronen bezeichnet die Anzahl der Neuronen in den beiden versteckten Schichten, mit der ersten, nach der Eingabeschicht befindlichen, beginnend.

# Erfolge (%)				**NRR**					
# Beob.	**1500**	**1000**	**600**	**200**	**150**	**100**	**80**	**60**	**40**
16×8 Neuronen	75	70	90	85					
32×16 Neuronen	100	97	94	94	92	74	68	36	6
Weight-Decay	100	94	86	42	18	6	6	4	2
2× # Lernschritte	100	100	90	85					
64×32 Neuronen	100	95	100	100					
2× # Lernschritte	85	80	90	80					
				KRR					
$\lambda = 0.025$	100	100	92	32	28	12	10	4	0
$\lambda = 0.05$	100	100	78	44	36	10	6	0	0
$\lambda = 0.075$	100	100	94	56	24	0	6	0	0
				Referenz					
# Beob.	**1800**	**1200**	**600**	**300**					
NFQ	100	100	88	46					
				RNRR auf POMDP					
# Beob.	**20000**	**10000**	**5000**	**2000**	**1000**				
32×16 Neuronen	75	40	45	5	5				

Die Tabn. 8.1 und 8.2 ermöglichen einen Vergleich der Anzahl der erfolgreichen Lernversuche, für die der Stab für mindestens 3000 bzw. 100000 Schritte balanciert werden kann. Oberhalb von 1500 beobachteten Transitionen kann der Stab bei Anwendung der NRR für die mittelgroße Netzarchitektur stets Policies liefern, die mindestens 3000 Schritte balancieren können. Mit 32×16 Neuronen können selbst bei sehr wenigen Daten noch häufig funktionierende Policies ermittelt werden. Unter Berücksichtigung von 100000 Balancierschritten erzielt die mittlere Netzarchitektur unter Anwendung von Weight-Decay bei mehr als 600 Beobachtungen die besten Ergebnisse, darunter jedoch fällt die Anzahl der erfolgreichen Versuche im Vergleich zum Einsatz ohne Weight-Decay schnell ab. Die KRR erreicht vergleichbare und stabilere Ergebnisse für mehr als 600 Beobachtungen. Unterhalb dieser Beobachtungszahl erzielt die NRR eine bessere Performance.

Im Vergleich zur NFQ sind sowohl NRR als auch KRR erfolgreich. Die erzielte Performance übersteigt die der NFQ. Bei einer höheren Zahl an Beobachtungen ab 600 liegt die Performance etwas höher, darunter sind die Policies deutlich performanter.

Die Anwendung der RNRR auf dem Cart-Pole-Problem als POMDP führte ebenfalls zu akzeptablen Ergebnissen. Dabei besteht der Beobachtungsraum ausschließlich aus dem Winkel des Stabes, nicht jedoch aus der Winkelgeschwindigkeit.

Tabelle 8.2: Anzahl der erfolgreichen Lernversuche (von insgesamt je 50) für lange Evaluationsläufe für das Cart-Pole-Problem. Ein Versuch wird als erfolgreich gewertet, falls der Stab mindestens 100000 Schritte balanciert werden kann. Weight-Decay bedeutet, dass die Zielfunktion um einen Weight-Decay-Term erweitert wird, 2× # Lernschritte heißt, dass doppelt so lange trainiert wurde, $x \times y$ Neuronen bezeichnet die Anzahl der Neuronen in den beiden versteckten Schichten, mit der ersten, nach der Eingabeschicht befindlichen, beginnend.

# Erfolge (%)					**NRR**				
# Beob.	**1500**	**1000**	**600**	**200**	**150**	**100**	**80**	**60**	**40**
16×8 Neuronen	75	65	85	75					
32×16 Neuronen	95	87	79	81	78	58	56	30	6
Weight-Decay	95	92	82	26	6	2	2	0	0
2× # Lernschritte	85	95	75	60					
64×32 Neuronen	85	85	75	90					
2× # Lernschritte	30	70	65	70					

Tabelle 8.3: Durchschnittliche Anzahl an Balancierschritten der Lernversuche (von insgesamt je 50) für das Cart-Pole-Problem. Nach 3000 Schritten wird der Versuch abgebrochen, so dass erfolgreiche Lernversuche mit ebensovielen Schritten in die Statistik eingehen. Die Abkürzung 2× heißt, dass doppelt so lange trainiert wurde, $x \times y$ Neuronen bezeichnet die Anzahl der Neuronen in den beiden versteckten Schichten, mit der ersten, nach der Eingabeschicht befindlichen, beginnend. Zum Vergleich sind Ergebnisse aus [57] eingetragen. Die Ergebnisse zeigen deutlich, dass die Rewards-Regression in beiden Ausprägungen (NRR und KRR) stabilere Ergebnisse erzielt als LSPI. KRR erreicht 100 Prozent erfolgreiche Policies ab 1000 Beobachtungen, NRR ab 1500 Beobachtungen, während LSPI bei 1500 Beobachtungen die Marke von 2350 Schritten erreicht.

	NRR		**LSPI**
# Beob.	$\mathbf{16 \times 8}$	$\mathbf{32 \times 16}$	
1500	2.3×10^3	3.0×10^3	2.35×10^3
1000	2.1×10^3	3.0×10^3	2.20×10^3
800	2.6×10^3	2.9×10^3	2.00×10^3
600	2.7×10^3	2.9×10^3	1.50×10^3
400	2.6×10^3	2.8×10^3	1.30×10^3
200	2.6×10^3	2.8×10^3	0.70×10^3
		NRR	
# Beob.	$\mathbf{32 \times 16}$, **2×**	$\mathbf{64 \times 32}$	$\mathbf{64 \times 32}$, **2×**
1500	3.0×10^3	3.0×10^3	2.7×10^3
1000	3.0×10^3	2.9×10^3	2.4×10^3
800	3.0×10^3	2.9×10^3	2.5×10^3
600	2.7×10^3	3.0×10^3	2.7×10^3
400	2.8×10^3	2.9×10^3	2.4×10^3
200	2.6×10^3	3.0×10^3	2.8×10^3
		KRR	
# Beob.	$\lambda = \mathbf{0.025}$	$\lambda = \mathbf{0.05}$	$\lambda = \mathbf{0.075}$
1500	3.0×10^3	3.0×10^3	3.0×10^3
1000	3.0×10^3	3.0×10^3	3.0×10^3
800	2.9×10^3	2.5×10^3	2.9×10^3
600	2.8×10^3	2.5×10^3	2.8×10^3
400	2.6×10^3	2.3×10^3	2.6×10^3
200	1.1×10^3	1.6×10^3	1.8×10^3

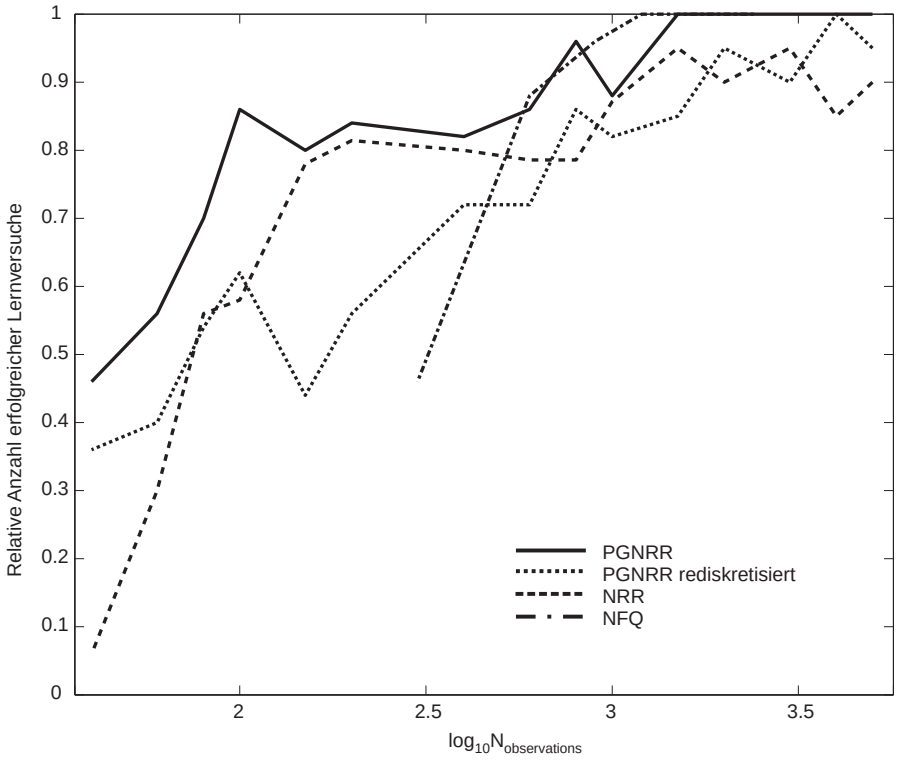


Abbildung 8.7: Erfolgreiche Lernversuche (von insgesamt je 50) auf dem Cart-Pole-Benchmark für NRR und PGNRR. Ein Versuch wird als erfolgreich gewertet, falls der Stab mindestens 100000 Schritte balanciert werden kann. Die rediskretisierte Variante der PGNRR ermöglicht einen besseren Vergleich mit der NRR, da die Mächtigkeit des Aktionsraumes eingeschränkt wird. Jedoch ist die PGNRR dann im Nachteil, weil die Policy nicht für diskrete Aktionen optimiert wurde. Zum Vergleich sind die Ergebnisse für die NFQ eingezeichnet, jedoch für 3000 Balancierschritte.

In Tab. 8.3 ist ergänzend die durchschnittliche Anzahl an Balancierschritten aufgeführt. Dies ermöglicht einen Vergleich zur LSPI. Es zeigt sich, dass in dieser Dimension die NRR und besonders die KRR Policies hervorbringen, die länger den Stab balancieren können als dies LSPI vermag. Unterhalb von 1500 Beobachtungen sind beide Verfahren deutlich besser.

Abb. 8.7 vergleicht die Anzahl der erfolgreichen Lernversuche von NRR und PGNRR untereinander und berücksichtigt dabei auch die NFQ. Dadurch wird die weitere Informationseffizienzsteigerung durch die aus der instantanen Adaption von Q-Funktion und Policy resultierende verschränkte Regularisierung von Q-Funktion und Policy demonstriert (siehe auch Abschn. 7.4.1). Die PGNRR erzielt demnach die besten und stabilsten Ergebnisse. Selbst die im Nachhinein diskretisierte Policy erreicht eine zur NRR vergleichbare Performance.

8.1.3.2 Wet-Chicken-Problem

Bezogen auf das Wet-Chicken-Problem dokumentieren Abb. 8.9 und Tab. 8.4 die hohe Informationseffizienz der Rewards-Regression. Unter Anwendung einer ε-greedy-Exploration (Abb. 8.9) erzielt die NRR nach einem Drittel der Zeit, die Prioritised-Sweeping benötigt, eine vergleichbare Performance. Q-Learning benötigt bereits eine Zehnerpotenz mehr Daten.

Prioritised-Sweeping arbeitet bereits mit einem Modell der Übergangswahrscheinlichkeiten, das auf allen beobachteten Daten beruht. Die Neural-Rewards-Regression verwendet zusätzlich neuronale Netze als mächtige Funktionsapproximatoren. Zusammen erklären diese Aspekte die hohe Informationseffizienz der NRR gegenüber Q-Learning.

Bei Anwendung von zufälliger Exploration von 500 Beobachtungen (Tab. 8.4) erzielt die KRR im Vergleich zu allen weiteren getesteten Verfahren die besten Ergebnisse mit Gauß-

Tabelle 8.4: Durchschnittlicher Reward für den Wet-Chicken-Benchmark nach 500 zufällig explorierten Beobachtungen für 1000 Evaluationsschritte unter Anwendung verschiedener Lernverfahren. Das Fuzzy-Reinforcement-Learning ist beschrieben in [5].

Verfahren	**øReward**
Kernel-Rewards-Regression	14.47 ± 0.07
Neural-Rewards-Regression	14.26 ± 0.07
Neural-Fitted-Q-Iteration	14.21 ± 0.07
Fitted-Value-Iteration mit lokal linearer Regression	14.17 ± 0.06
Fitted-Value-Iteration auf Clustern	13.92 ± 0.05
Fuzzy-Reinforcement-Learning	13.88 ± 0.02
DP mit anschließender linearer Regression	13.72 ± 0.05
Dynamische Programmierung	12.77 ± 0.05

Kernels mit $\sigma = 2.5$ und $\lambda = 0.5$. Die NRR liegt auf Platz 2 vor der NFQ (jedoch im gleichen Signifikanzbereich). Da der Lernprozess durch die verschränkte Iteration darüber hinaus schneller abläuft als der der NFQ, ist die NRR vorzuziehen. Der ebenfalls mächtige lokallineare Ansatz der Fitted-Value-Iteration liegt auch im entsprechenden Signifikanzbereich.

In der Abb. 8.8 sind ferner zum Nachweis der Funktionsweise der Explicit-Kernel-Rewards-Regression Ergebnisse auf dem einfachen Wet-Chicken-Problem dargestellt. Ohne die über die Ridge-Regression hinaus gehende Regularisierung liefert die EKRR die gleichen Ergebnisse wie die KRR, falls die stark optimale Policy eindeutig ist. Erwartungsgemäß führt das Anwenden der Regularisierungsmaßnahmen Minimieren des Gewichtsvektors und Berücksichtigen eines ε-Schlauches zu konservativeren Policies.

8.1.3.3 Rekonstruierbarkeit von Markov-Zuständen

In Tab. 8.5 ist die Rekonstruierbarkeit der beiden Zustandsvariablen des Cart-Pole-Problems (Winkel und Winkelgeschwindigkeit) an Hand des internen Zustandes angegeben. Dies soll als Nachweis dafür dienen, dass die Markov-Eigenschaft tatsächlich hergestellt werden kann.

Dazu wurden die Korrelationen für die jeweils bestmöglichen Linearkombinationen der internen Zustandsvariablen zur Beschreibung der externen Zustandsvariablen (im Least-Squares-Sinn) mit ebendiesen bestimmt.

Tabelle 8.5: Stärkste Korrelationen zwischen internen Zuständen $\mathbf{x}$ and externen Zuständen s für das partiell beobachtbare Wet-Chicken- und das Cart-Pole-Problem. Beim Wet-Chicken-Problem wurde nur die Position des Kanus s und beim zweidimensionalen Cart-Pole-Problem der Winkel des Stabes beobachtet. Die Einträge beschreiben, wie gut auch die nicht beobachtbaren Zuständen von den internen Zuständen abgebildet werden können. Dazu wurden die Korrelationen für die jeweils bestmöglichen Linearkombinationen der internen Zustandsvariablen mit den externen, auch nicht beobachtbaren Zustandsvariablen herangezogen. Die beste Linearkombination sei dann erzielt, wenn der quadratische Fehler zum externen Zustand minimal ist.

Benchmark	**Var.**	**Beob.**	**Korr.**	**Benchmark**	**Var.**	**Beob.**	**Korr.**
Cart-Pole	θ	ja	0.9977	Wet-Chicken	s	ja	0.9996
	$\dot{\theta}$	nein	0.9896		t	nein	0.6552

Die Tabelle zeigt darüber hinaus die Rekonstruierbarkeit der Zustandsvariablen des partiell beobachtbaren Wet-Chicken-Problems, bei dem nur die Position des Kanus sichtbar gemacht wird. Bis auf die Strömungsvariable t, deren Korrelation zur tatsächlichen Größe 0.66 beträgt, können alle Zustandsvariablen nahezu fehlerfrei rekonstruiert werden. Der interne Zustandsraum besteht in beiden Fällen aus vier Variablen.

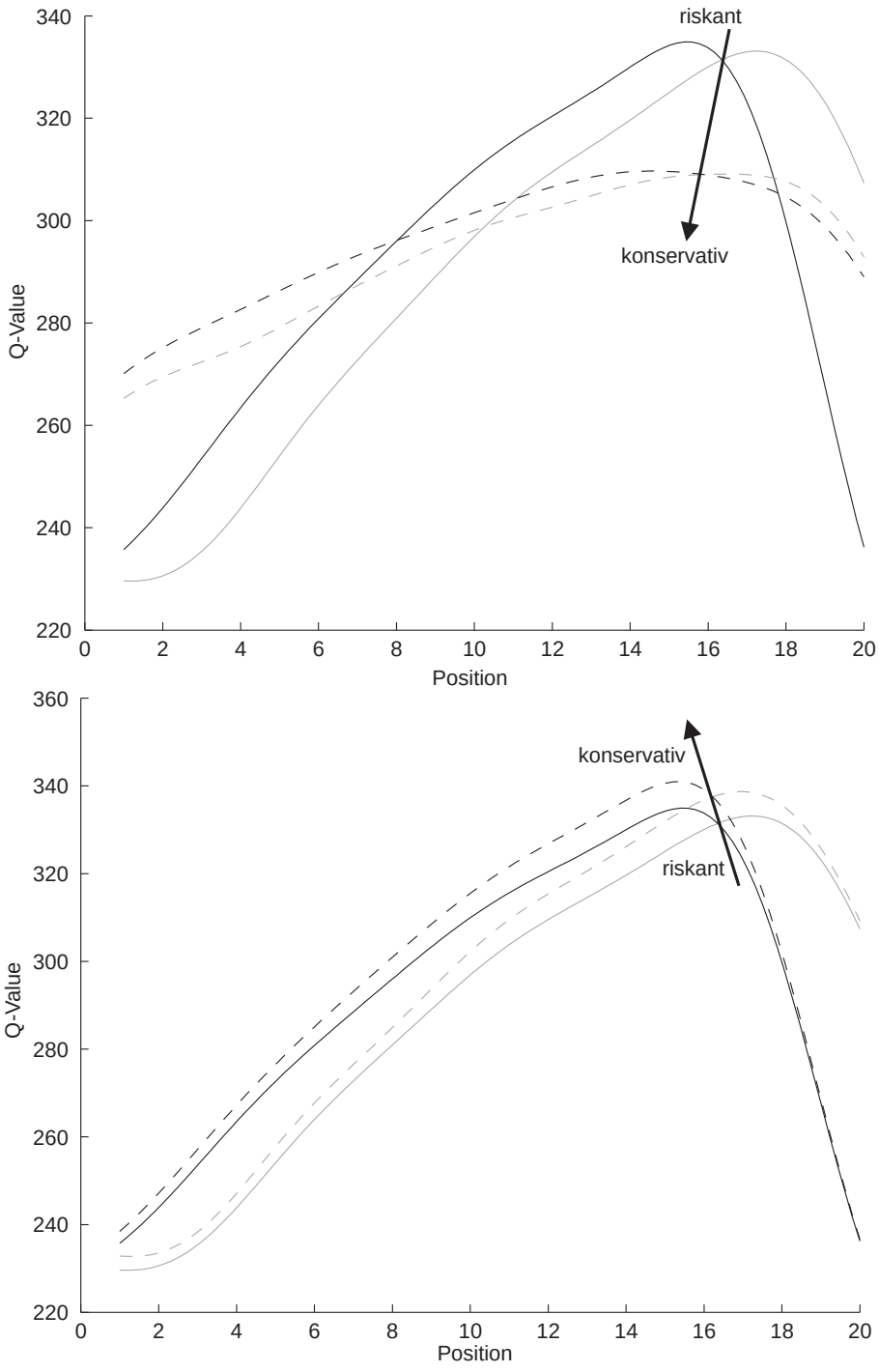


Abbildung 8.8: Exemplarische Q-Funktionen für das Wet-Chicken-Problem unter Anwendung der Explicit-Kernel-Rewards-Regression. Während sich die durchgezogene Linie jeweils auf die Ridge-KRR-Lösung bezieht, betreffen die gestrichelten Q-Funktionen, die als Lösungen der EKRR mit zusätzlicher Regularisierung, die in der Minimierung des Gewichtsvektors (oben) und im Berücksichtigen eines ε-Schlauches (unten) bestehen, hervorgehen. Es zeigt sich, dass mit stärkerer Regularisierung die ermittelten Policies konservativer sind, da der Schnittpunkt der Aktionen Zurückrudern und Treiben weiter nach links verschoben wird.

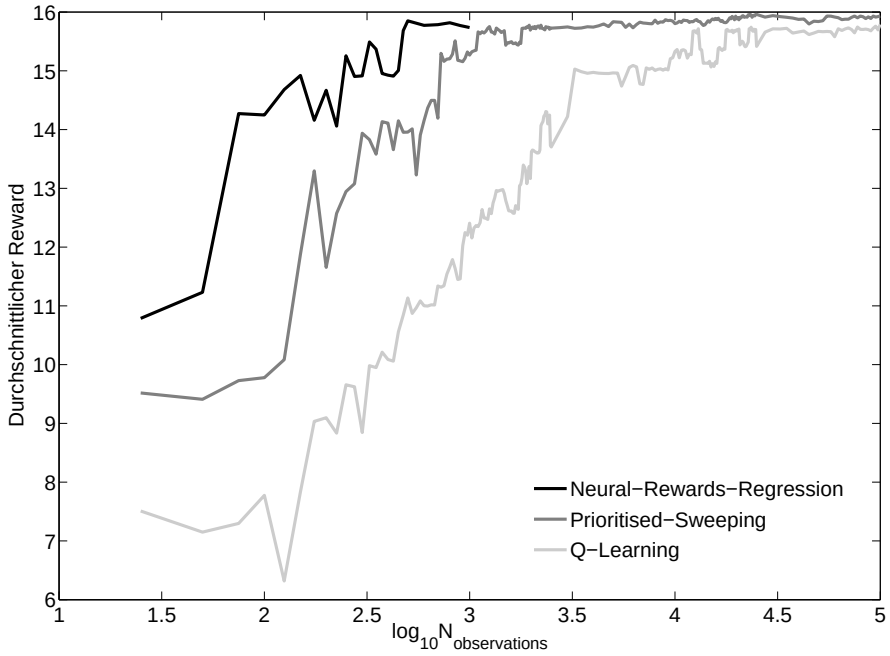


Abbildung 8.9: Durchschnittlich erzielter Reward auf dem Wet-Chicken-Benchmark für die NRR im Vergleich zu Prioritised-Sweeping und Q-Learning mit ε-greedy-Exploration. Dabei wird $\varepsilon = \frac{1}{3}$ gewählt. Prioritised-Sweeping ist bereits um eine Größenordnung dateneffizienter als Q-Learning, während die NRR die Informationseffizienz nochmals um eine Größenordnung verbessert.

8.2 Industrielle Anwendungen

Im Rahmen unterschiedlicher Projekte zur Regelung von Gasturbinen-Simulationen als Vorbereitung auf den Einsatz von RL auf realen Gasturbinen [81] wurde die Neural-Rewards-Regression zur Meta-Steuerung mit dem Ziel einer Performanceverbesserung eingesetzt. Wir betrachten dabei zwei Projekte, bei denen die hier so bezeichneten Brennkammer- und Emissionsreduktions-Simulationen geregelt wurden. Die Bezeichnungen der Gasturbinen-Simulationen beziehen sich auf die Zielsetzung der jeweiligen Projekte. In beiden Fällen sind neben vielen weiteren Elementen der Gasturbine sowohl die Brennkammer als auch das Emissionsverhalten Bestandteil der Simulation.

8.2.1 Herausforderungen bei der Steuerung von Gasturbinen

Gasturbinen sind, im Gegensatz zu Stein- und Braunkohlekraftwerken, umweltfreundliche Erzeuger elektrischer Energie. Darüber hinaus ist ihre Effizienz bedeutend höher. In Kombination mit angeschlossenen Dampfturbinen (Gas- und Dampfturbinen-Anlagen (GuD)) kann ein Wirkungsgrad von über 60 Prozent erzielt und die Emissionslast weiter reduziert werden. GuD-Kraftwerke emittieren bei gleicher Leistung bis zu 50 Prozent weniger Klimagase als moderne Kohlekraftwerke [81]. Durch die kurze Anlaufzeit und die Fähigkeit schneller Leistungsänderungen sind Gasturbinen außerdem geeignet, die Fluktuationen von alternativen Energieerzeugern wie Wind- und Solarkraftwerken auszugleichen. Sie können mit Erdgas und Erdöl verschiedener Qualität betrieben werden.

Ziele der aktuellen Entwicklungen bestehen vor allem in der weiteren Reduktion von Emissionen wie etwa Stickstoffoxiden (NO_X), Kohlenmonoxid (CO) und unverbrannten Koh-

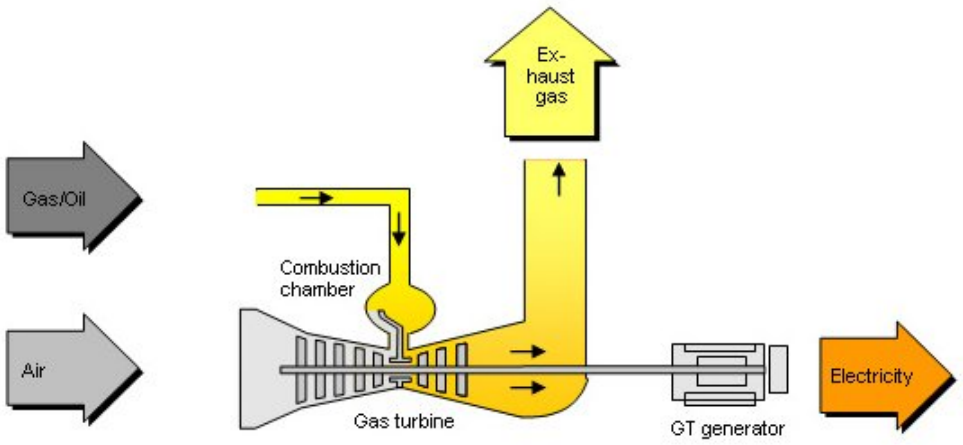


Abbildung 8.10: Schematische Darstellung einer Gasturbine und ihrer Funktionsweise.

lenwasserstoffen [92]. Die Reduktionsmöglichkeiten für Schwefeloxide (SO_X) sind bereits weitgehend ausgereizt. Dabei soll eine hohe Effizienz natürlich aufrechterhalten bleiben oder eine durch das Stromnetz vorgegebene Last stabil gehalten werden. Ein weiteres Problem besteht in der Dynamik des Verbrennungsprozesses in der Brennkammer (Brennkammerdynamik), die im akustisch wahrnehmbaren Frequenzbereich liegt. Unter schwer vorhersehbaren Bedingungen kann die Dynamik so stark werden, dass die Leistung der Turbine reduziert oder diese sogar abgeschaltet werden muss. Die Prognose und Beherrschbarkeit dieser Dynamik ist von großer Bedeutung für einen stabilen Betrieb.

Um diese Ziele zu erreichen, müssen verschiedene Mess- und Stellgrößen der Turbinen in optimaler Weise aufeinander abgestimmt werden. Zu ihnen gehören Gasströme, Temperaturen, Drücke und Umweltbedingungen. Neben instantanen Einflüssen müssen auch mittelfristige dynamische Effekte berücksichtigt werden. Daher kann Reinforcement-Learning zur Regelung von Gasturbinen in Betracht gezogen werden.

8.2.2 Die Gasturbinen-Simulationen

Unsere Simulationen realer Gasturbinen beruhen auf rekurrenten neuronalen Netzen. In ihrer Skizzierung kann eine detaillierte Beschreibung aus Gründen der Vertraulichkeit nicht erfolgen. Aus diesem Grund sind die Parameter, Variablen und Ergebnisse normiert und daher dimensionslos aufgeführt.

8.2.2.1 Erste Simulation – Minimierung der Brennkammerdynamik

Beim Brennkammerdynamik-Projekt ging es vornehmlich um die Reduktion der Brennkammerdynamik. Die verwendeten Daten basieren auf Monitoring-Datenbänken einer realen Gasturbine des entsprechenden Typs, die für die Energieerzeugung im operativen Betrieb eingesetzt wird. Es handelt sich um insgesamt 1.26 Millionen Datensätze im Zeitraum vom 22.08.2005 bis zum 26.01.2006, die im 5-Sekunden-Raster aufgenommen wurden. Mit Hilfe unterschiedlicher statistischer Analyseverfahren (Experteninformationen, Korrelationsanalyse, Bayes-Netze [67]) wurden aus insgesamt über 2000 Attributen etwa 100 Parameter ausgewählt, die durch eine Sensitivitätsanalyse eines rekurrenten neuronalen Netzes und Hinzufügen weiterer Rechengrößen auf 20 Attribute reduziert wurden. Dazu gehören neben der Leistung die Umgebungs-, Gas- und Abgastemperatur, der Umgebungsluft-, Gas- und Abgasdruck, Druckdifferenzen, die relative Umgebungsluftfeuchtigkeit, die Vorleitschaufel- und Pilotgasventilstellung sowie verschiedene Brennkammerdynamikgrößen und weitere von Experten definierte abhängige Größen.

Die insgesamt verfügbaren Daten wurden so angereichert, dass 3 Minuten vor und 1.5 Minuten nach Triggerereignissen berücksichtigt und genauso viele Datensätze im Normalbetrieb hinzugefügt wurden. Darüber hinaus müssen Bedingungen an Betriebsparameter und Messgrößen erfüllt sein, die den Lastbetrieb der Turbine charakterisieren. Triggerereignisse werden zu Zeitpunkten erzeugt, an denen die Brennkammerdynamik bestimmte vorgesehene Limits überschreitet.

Die angereicherte Datenbasis besteht schließlich noch aus ca. 78000 Datensätzen, mit denen die Simulation der Gasturbine mit einem speziellen rekurrenten neuronalen Netz trainiert wurde. Attribute, die sich instantan aus Steuergrößen und Umwelteinflüssen ergeben bzw. für die diese Eigenschaft vermutet wird, werden dabei mit einem statischen Feed-Forward-Netz bestimmt und dann neben den dynamischen Größen als Eingabe für ein rekurrentes Netz verwendet, das nur die dynamischen Größen prognostiziert. Leistung, Gas- und Abgastemperatur sowie Abgasdruck sind etwa statische, die Druckdifferenz am Ansaugfilter und die Brennkammerdynamikgrößen dynamische Größen. Umgebungsgrößen wie etwa die Umgebungstemperatur und der -luftdruck werden wie dynamische Größen behandelt, können und müssen jedoch nicht prognostiziert werden, da sie praktisch nicht vom Betrieb der Gasturbine abhängen.

Der Aktionsraum der Simulation besteht schließlich aus den beiden Steuerparametern Vorleitschaufelstellung (IGV) und Pilotgasventilstellung (Pilot), das die Brennstoffzufuhr der Pilotflamme regelt. Qualitativ besteht eine hohe positive Korrelation zwischen dem IGV und der Leistung der Turbine, das Erhöhen von Pilot führt zu einer moderaten Reduktion der Brennkammerdynamik.

8.2.2.2 Zweite Simulation – Minimierung der Emissionslast

Die Emissionsreduktions-Simulation bezieht sich auf einen anderen Turbinentyp. Qualitativ sind die Datenaufbereitungs- und Analyseschritte den oben beschriebenen sehr ähnlich. Die Anzahl der letztlich zu betrachtenden Attribute beläuft sich hier jedoch auf 28. Der Aktionsraum vergrößert sich um die Regulierung der Brennstoffzufuhr in verschiedene Brennstoffströme, zu denen nach wie vor Pilot gehört. Trotz der Einflussmöglichkeiten auf IGV wurde diese Steuergröße unberücksichtigt und konstant gelassen, da das Ziel des Projektes darin bestand, den Einfluss der Brennstoffaufteilung zu optimieren, um neben der Beherrschung der Brennkammerdynamik eine weitere Reduktion der Emissionen, insbesondere den Ausstoß von Stickstoffoxiden zu erreichen.

8.2.3 Neural-Rewards-Regression zur Steuerung der Gasturbinen-Simulationen

Für beide Simulationen hat sich herausgestellt, dass Reinforcement-Learning auf dem Zustandsraum selbst nicht erfolgversprechend ist, offenbar erfüllt dieser nicht die Markov-Bedingung. Da die Recurrent-Neural-Rewards-Regression jedoch MDP höherer Ordnung und damit bis zu einem bestimmten Grad partiell beobachtbare MDP beherrschen kann, ziehen wir zum Vergleich Standard-Verfahren heran, die auf einem internen Zustand, wie auch die RNRR, arbeiten. Dieser interne Zustand wird mit Hilfe eines rekurrenten Netzes bestimmt, dabei wird eine Reduktion der Dimension des internen Zustandes angestrebt, wie dies auch bei der RNRR der Fall ist (siehe Kap. 7.5).

Wir betrachten also neben dem Referenz-Controller, repräsentiert durch die Datenbasis, der RNRR und der RPGNRR, TD-Learning und vollständige Policy-Iteration auf dem internen Zustand des zuvor bestimmten rekurrenten Netzes. Diese Methoden bezeichnen wir als RQ (Recurrent-Q-Learning) und RPS (Recurrent-Prioritised-Sweeping), was die konkrete Realisierung der jeweiligen Verfahren zum Ausdruck bringt. Einzelheiten dazu sind in [92] beschrieben.

Für die Optimierung wurden je nach Projektziel unterschiedliche Reward-Funktionen verwendet, die in mehreren Iterationen den qualitativen Zielsetzungen angepasst wurden. In bei-

den Fällen ist der Reward ausschließlich vom Folgezustand abhängig. Für die Brennkammer-Simulation ist er definiert als

$$R(s') = \frac{\mathrm{PEL} - \mathrm{PEL}_{\max}}{\mathrm{PEL}_c} - 2[\mathrm{RMS} - \mathrm{RMS}_{\mathrm{limit}}]_+^2 + b. \tag{8.26}$$

Dabei bezeichnen PEL die Leistung im Zustand s', $\mathrm{PEL}_{\max}$ die Maximalleistung, PEL_c eine Normleistung, RMS die Brennkammerdynamik im Zustand s', $\mathrm{RMS}_{\mathrm{limit}}$ die tolerierbare Brennkammerdynamik und b einen Offset. Für die Emissionsreduktions-Simulation wurden mehrere Dynamik-Bänder betrachtet, darüber hinaus ist der Emissionswert der Stickstoffoxide ein Argument der Reward-Funktion, die sich als

$$R(s') = \frac{r_{\mathrm{PEL}} - r_{\mathrm{NOX}} - r_{\mathrm{RMS}} + b}{c} \tag{8.27}$$

$$r_{\mathrm{PEL}} = \mathrm{PEL} \tag{8.28}$$

$$r_{\mathrm{NOX}} = \frac{\mathrm{NOX}}{\mathrm{NOX}_{c_1}} + \frac{\mathrm{logistic}\left(\frac{1}{2}(\mathrm{NOX} - \mathrm{NOX}_{\mathrm{limit}})\right)}{\mathrm{NOX}_{c_2}} \tag{8.29}$$

$$r_{\mathrm{RMS}} = \sum_{i=1}^{n_{\mathrm{RMS}}} \frac{[\mathrm{RMS}_i - \mathrm{RMS}_{i,\mathrm{limit}}]_+^2}{\mathrm{RMS}_{i,c}} \tag{8.30}$$

darstellen lässt. Es bezeichnen zusätzlich NOX die Emissionsmenge von Stickstoffoxiden im Zustand s', $\mathrm{NOX}_{\mathrm{limit}}$ das tolerierbare Limit, NOX_{c_1} und NOX_{c_2} Normemissionen sowie $\mathrm{RMS}_{\cdot,c}$ Normdynamikgrößen. Der Index für die RMS-Werte bezeichnet die verschiedenen Brennkammerdynamikbänder. Die Reward-Funktion ist in beiden Fällen so festgelegt, dass im Regelbetrieb $R(s')$ zwischen 0 und 1 liegt.

8.2.4 Ergebnisse

Für die RNRR erzielten wir das beste Ergebnis mit Q-Funktionen, deren Netzarchitektur aus 16 Neuronen in der ersten und 8 Neuronen in der zweiten Schicht bestehen. Gemäß unserer Beobachtung zur verschränkten Regularisierung, erhöhten wir die Komplexität der Q-Funktions-Netze für die RPGNRR auf 30 und 15 Neuronen und wählten eine weniger komplexe Struktur, 10 und 5 Neuronen, für das Policy-Netzwerk. Als Lernalgorithmus verwendeten wir das Vario-Eta-Verfahren [68]. Wir wählten eine Batch-Size von 5 und $\eta = 0.05$ sowie einen Weight-Decay-Faktor von $\lambda = 0.001$. Die Eingabedaten wurden varianznormiert und das Training mit einer exponentiell abfallenden Lernrate durchgeführt.

Tabelle 8.6: Durchschnittlicher und finaler Reward bei Anwendung der verschiedenen Controller auf den beiden Gasturbinen-Simulationen. Die Ergebnisse beziehen sich auf die Graphen in Abb. 8.11 und fassen die gesamte durchschnittliche Performance und die durchschnittliche Performance an den angestrebten Arbeitspunkten zusammen.

Methode	Brennkammer-Simulation		Emissionsreduktions-Simulation	
	Mittelwert	Final	Mittelwert	Final
RefCon	0.53 ± 0.01	0.53 ± 0.01	0.23 ± 0.02	0.23 ± 0.02
RQ	0.72 ± 0.01	0.73 ± 0.01	0.29 ± 0.04	0.29 ± 0.04
RPS	0.839 ± 0.005	0.842 ± 0.005	0.45 ± 0.03	0.51 ± 0.04
RNRR	0.851 ± 0.004	**0.854 ± 0.005**	**0.60 ± 0.04**	0.63 ± 0.04
RPGNRR	**0.861 ± 0.004**	**0.862 ± 0.004**	**0.61 ± 0.04**	**0.69 ± 0.04**

Die in Tab. 8.6 dargestellten Ergebnisse zeigen den durchschnittlich erzielten Reward über 1000 Evaluations-Trajektorien, gemittelt über die Zeit und den finalen Wert. Jeder Versuch startet von einem der tatsächlich auf der Turbine gemessenen Zustände, mit denen die Simulation trainiert wurde. Die fettgedruckten Rewards liegen innerhalb des Signifikanzbereiches des jeweils maximalen Wertes.

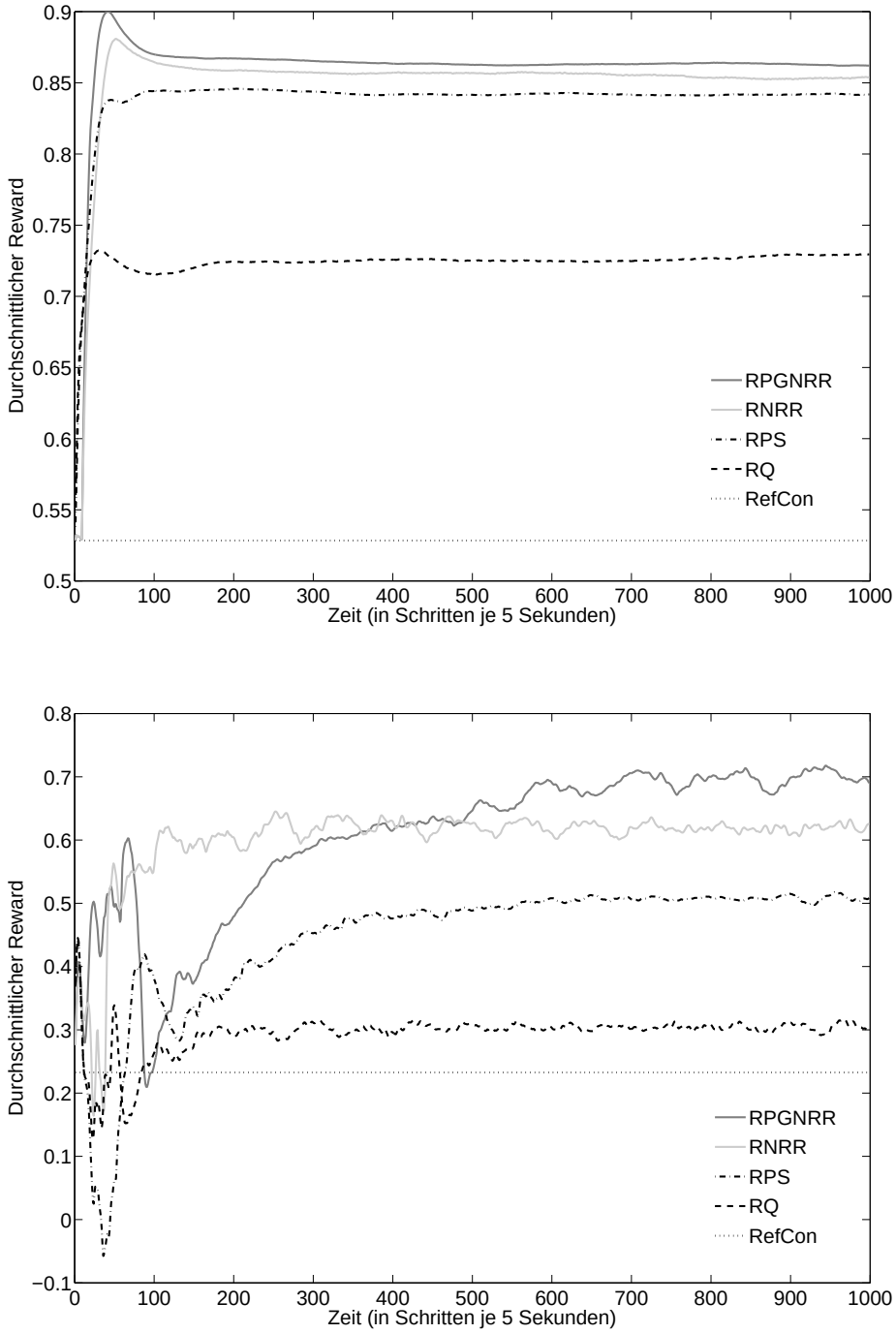


Abbildung 8.11: Reward-Entwicklung für die verschiedenen betrachteten Controller auf der Brennkammer-Simulation (oben) sowie auf der Emissionsreduktions-Simulation (unten) über 5000 Sekunden für Referenz-Controller (RefCon), TD-Learning (RQ) und Policy-Iteration auf internem Zustandsraum (RPS), Recurrent-Neural-Rewards-Regression (RNRR) sowie Recurrent-Policy-Gradient-Neural-Rewards-Regression (RPGNRR). Die Ergebnisse sind gemittelt über 1000 unterschiedliche Startzustände.

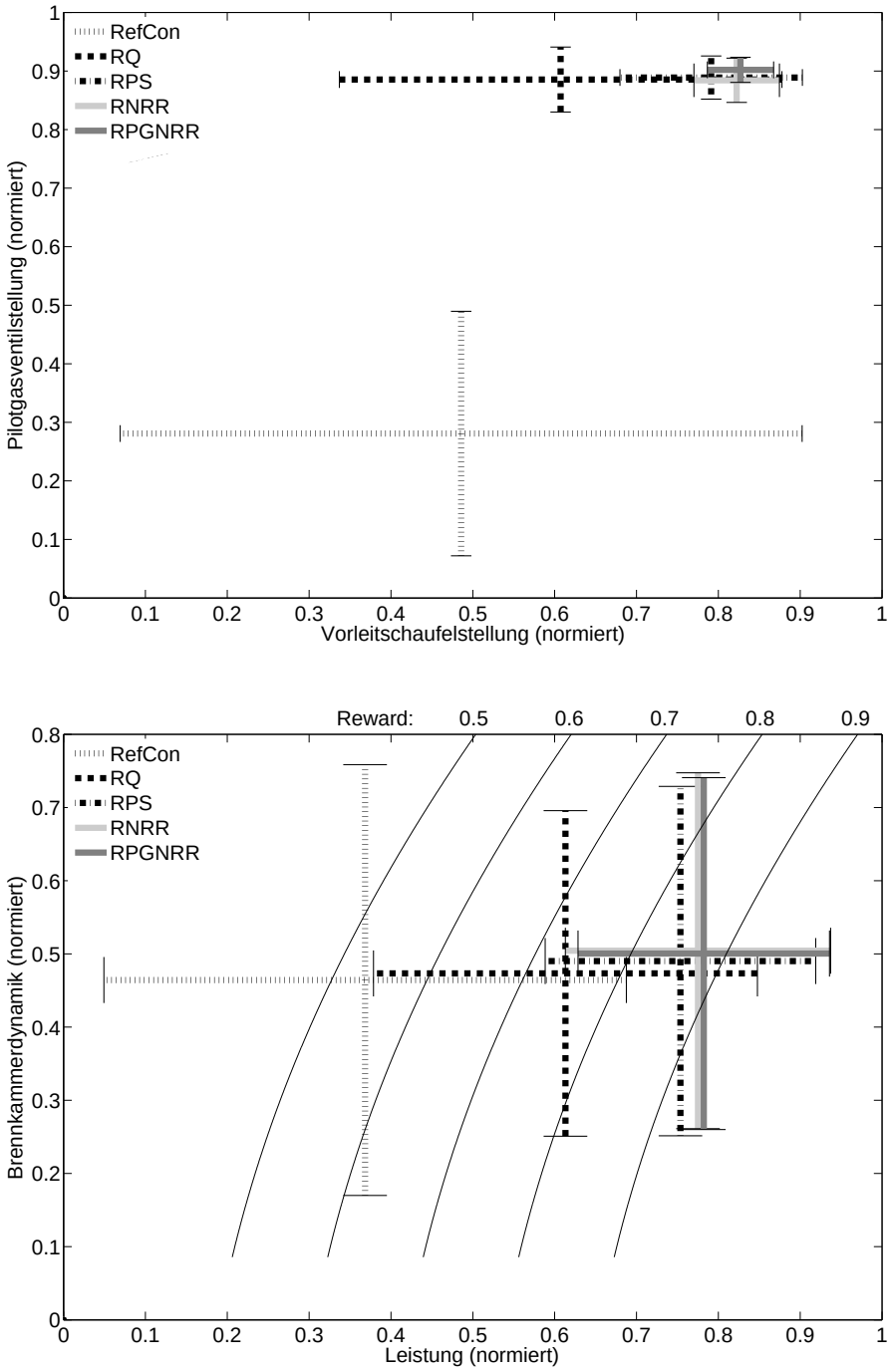


Abbildung 8.12: Vergleich der Stellgrößen IGV und Pilot (oben) sowie der Messgrößen Leistung und RMS (unten) und ihre Standardabweichung an den durchschnittlichen Arbeitspunkten, die von verschiedenen Controllern erreicht werden, auf der Brennkammer-Simulation. “Iso-Reward-Kurven” indizieren identische Rewards innerhalb dieser Projektion im Zustandsraum.

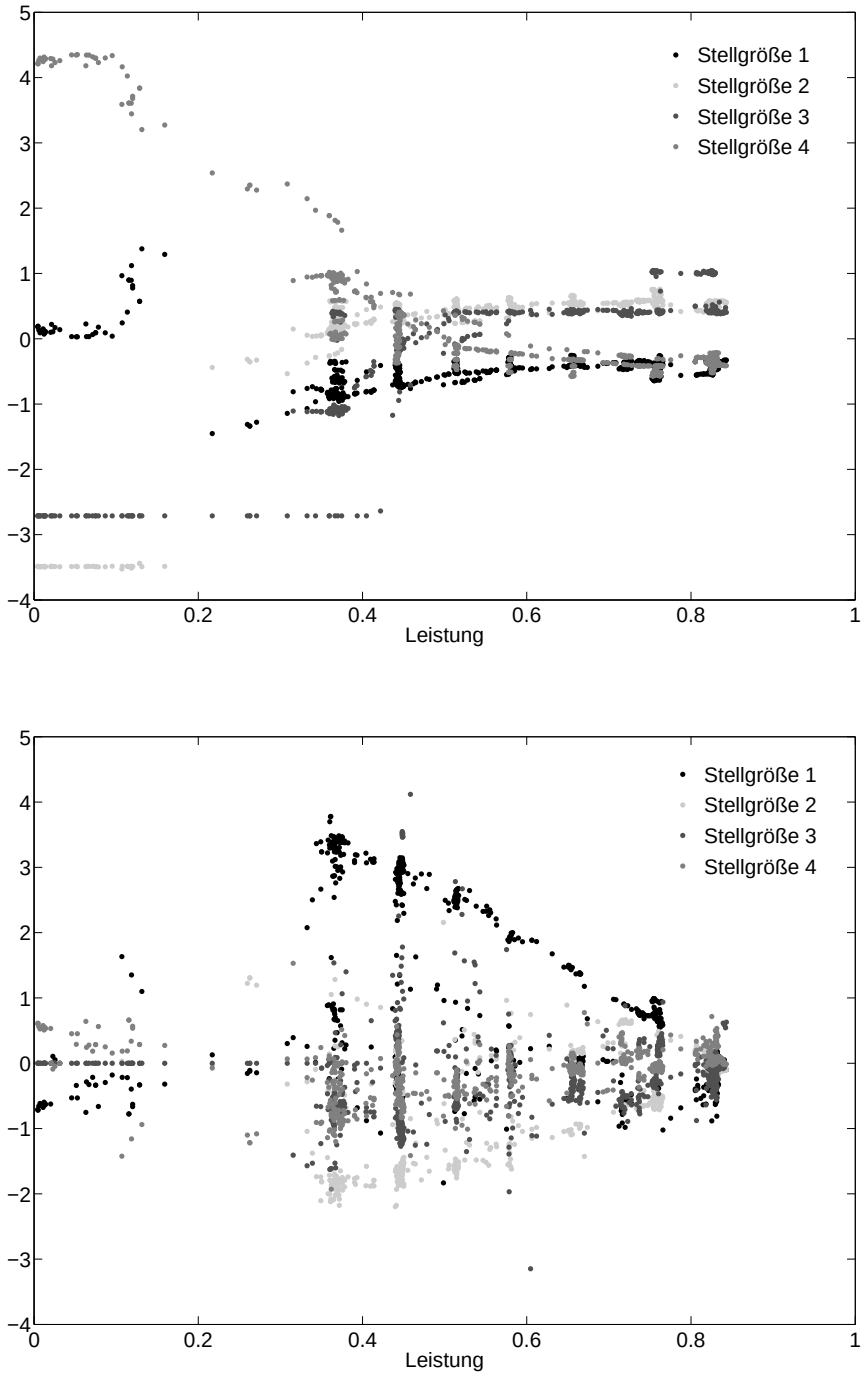


Abbildung 8.13: Initiales Setting (oben) und Differenz zwischen initialem und finalem Setting (unten) ausgewählter Stellgrößen für die Emissionsreduktions-Simulation in Abhängigkeit von der Leistung der Gasturbine nach Anwendung der RNRR. Die initialen Werte wurden aus der Menge aller auf der echten Gasturbine während der Exploration gemessenen Größen gesampelt. Die Graphen verdeutlichen, dass für eine Verbesserung der Performance die Stellgrößen 1 und 2 stark gespreizt werden sollten. Dies stellt eine wesentliche Empfehlung zur Änderung des Arbeitspunktes dar.

RPGNRR und RNRR erreichen jeweils die höchste Performance. Da diese an Hand der oben beschriebenen Reward-Funktionen optimiert wird, die der Referenz-Controller nicht kennt, sind alle anderen Verfahren besser als der Referenz-Controller. In der einfacheren Brennkammer-Simulation erreicht Policy-Iteration bereits bemerkenswerte Ergebnisse, während die Unterschiede zu den Verfahren RNRR und RPGNRR bei der Emissionsreduktions-Simulation deutlich größer sind.

Die Abb. 8.11 zeigt den mittleren Reward-Verlauf während der Evaluation. Bei der Brennkammer-Simulation kommt es zu einer zügigen Stabilisierung der angestrebten Arbeitspunkte. Die Verfahren RPGNRR und RNRR erzielen den höchsten durchschnittlichen Reward und erreichen die besten Arbeitspunkte. Die Dynamik der Emissionsreduktions-Simulation ist komplexer, stabile Arbeitspunkte können erst nach mehreren Simulations-Iterationen erreicht werden. Auch hier erzielen RPGNRR und RNRR nach einer entsprechenden Anzahl an Simulationsschritten den höchsten durchschnittlichen Reward.

8.2.4.1 Neue Arbeitspunkte für die Brennkammer-Simulation

Für die Brennkammer-Simulation haben wir die angestrebten Arbeitspunkte untersucht. Die Abb. 8.12 zeigt die angefahrenen Steuergrößen und die für die Berechnung der Reward-Funktion relevanten Messgrößen. Bzgl. der Steuerparameter weisen RNRR und RPGNRR die größte Bestimmtheit auf, während RQ, RPS und der Referenz-Controller insbesondere im IGV stark streuen. Bzgl. der Leistung und der Brennkammerdynamik unterscheiden sich die Standardabweichungen nicht wesentlich, im Erwartungswert geht jedoch eine Erhöhung der Leistung auch mit einer Erhöhung der Brennkammerdynamik einher. Die "Iso-Reward-Kurven" verdeutlichen die Auswirkungen auf die Rewards. Sie zeigen, dass trotz leicht erhöhter Brennkammerdynamik bei Anwendung der RPGNRR und der RNRR die Leistung soweit ansteigt, dass der Reward erhöht werden kann.

8.2.4.2 Neue Arbeitspunkte für die Emissionsreduktions-Simulation

Bei der Emissionsreduktions-Simulation sind durch die Anwendung der RNRR neue Erkenntnisse über vorteilhafte Arbeitspunkte gewonnen worden. Die Abb. 8.13 zeigt vier Stellgrößen von 1000 zufällig gewählten Arbeitspunkten der Gasturbine für den Referenz-Controller sowie ihre Differenz zu den Stellgrößen des von der RNRR ermittelten Controllers. Die Namen der Stellgrößen können aus Gründen der Vertraulichkeit nicht genannt werden, außerdem sind die Werte selbst mittelwert- und varianznormiert. Es zeigt sich deutlich, dass zwar die Stellgrößen 3 und 4 um die Einstellungen des Referenz-Controllers liegen, jedoch die Stellgrößen 1 und 2, auch im Mittelwert, erheblich davon abweichen. Dieser Vorschlag wurde nach den hier beschriebenen Experimenten bei den weiteren Betrachtungen des Auftraggebers mit großem Interesse ins Kalkül gezogen, da er auch aus technisch-physikalischer Motivation heraus zeitgleich ähnliche Vermutungen anstellte. Inzwischen wird diese Erkenntnis für den operativen Betrieb der Gasturbinen berücksichtigt.

Teil III

Bayes'sche Perspektive – Generalisierung und Qualitätssicherung durch Unsicherheitspropagation

Kapitel 9

Unsicherheitspropagation und Bellman-Iteration

Dieser Teil beschäftigt sich mit Generalisierung und damit Steigerung der Informationseffizienz auf der Grundlage der Bayes'schen Philosophie. Wir nutzen dazu das Verfahren der Unsicherheitspropagation, das zur Analyse von statistischen Unsicherheiten in physikalischen Systemen verwendet wird und dann zum Einsatz kommt, wenn man an der Unsicherheit von Ergebnissen interessiert ist, die analytisch von unsicherheitsbehafteten Messgrößen abhängen.

9.1 Übersicht

Wir wenden die Unsicherheitspropagation dabei auf die Fitted-Value-Iteration oder, im diskreten Fall, auf die dynamische Programmierung an [102]. Auf diese Weise gewinnen wir, ausgehend von der angenommenen Verteilung über die Modellparameter, schließlich eine Verteilung über MDP, die zu einer Verteilung über Performances oder Q-Funktionen für verschiedene Policies führt. Dabei kann man sich (zunächst) auf die Parameter Erwartungswert und Standardabweichung beschränken, da diese hinreichend zur Beschreibung von erwarteter Performance und ihrer Unsicherheit sind.

9.1.1 Informationseffizienz und Qualitätssicherung

Ein wesentlicher Aspekt neben der Informationseffizienzsteigerung besteht in der Qualitätssicherung bei der Anwendung von Reinforcement-Learning. Diese beiden Zielsetzungen sind eng miteinander verknüpft, da mit wachsender Anzahl an Beobachtungen die Unsicherheit fällt. Durch die explizite Bestimmung der Unsicherheit der ermittelten Q-Funktionen wird die Angabe einer unteren Schranke für die erzielbare Performance mit einer zuvor festgelegten Wahrscheinlichkeit ermöglicht. Diese sogenannte garantierte Performance (mit einer bestimmten Wahrscheinlichkeit) ist demzufolge gerade umso größer, je kleiner die Unsicherheit ist. Sie kann direkt als ein Kriterium zur Qualitätssicherung herangezogen werden.

Neben der Bestimmung der minimal garantierten Performance besteht darüber hinaus das Ziel, diese anzuheben und zu maximieren. Wir erhalten auf diese Weise schließlich sichere und sicherheitsoptimale Policies, die sich von den klassischen erwartungswert-optimalen Policies unterscheiden, indem sie die Performance an einem bestimmten Perzentil bzgl. der Verteilung über MDP optimieren.

9.1.2 Methode

Um die oben beschriebenen Ziele zu erreichen, beschäftigen wir uns mit der Determinierung, Quantisierung und Minimierung der Unsicherheiten von Schlussfolgerungen aus Messwer-

ten im Reinforcement-Learning, also den Unsicherheiten von Q-Funktionen und Policies. In diesem Zusammenhang leiten wir die Unsicherheitspropagation für Reinforcement-Learning (UPRL) für diskrete MDP und für die Least-Squares-Policy-Iteration [57] her und gehen auf die Anwendbarkeit im Rahmen der frequentistischen und der Bayes'schen Perspektive ein.

Wir werden darüber hinaus zwischen lokaler und globaler Sicherheitsoptimalität unterscheiden und schließlich außer der Schätzung der Q-Werte und ihrer Unsicherheiten höhere statistische Momente wie die Schiefe und die Kurtosis sowie die Kovarianzen und entsprechend die Koschiefe und Kokurtosis zwischen verschiedenen Q-Werten ins Kalkül ziehen.

In der unabhängigen Arbeit [22] von Delage und Mannor wurde ein ähnliches Problem gelöst, jedoch mit Methoden, die auf konvexer Programmierung beruhen. Das in der vorliegenden Arbeit vorgestellte Verfahren ist jedoch in einem allgemeineren Kontext hinsichtlich sowohl der verwendeten Funktionsapproximatoren als auch der statistischen Paradigmen einsetzbar.

9.2 Unsicherheitspropagation für diskrete MDP

Die Ursache von Unsicherheit im Reinforcement-Learning ist das Unwissen oder lückenhafte Wissen über die wahre Umgebung, den wahren MDP. Je mehr Beobachtungen gemacht werden, desto sicherer kann sich der Beobachter über den MDP sein. Aber je größer die Stochastizität, desto mehr Unsicherheit über den MDP bleibt für eine bestimmte Anzahl an Beobachtungen bestehen. Tatsächlich geht aus einer Beobachtung eines Zustands-Aktions-Paares das vollständige Wissen über dieses Paar hervor, falls der MDP bekanntermaßen vollständig deterministisch ist. Ist das System jedoch stark stochastisch, dann besteht ein entsprechend hohes Risiko, einen niedrigen erwarteten Gesamtertrag zu erhalten.

Aus dem Zusammenspiel zwischen der Anzahl der Beobachtungen und der Stochastizität des MDP folgt, dass die Unsicherheit sich qualitativ von der Stochastizität des MDP unterscheidet, die zu dem Risiko, einen niedrigen Gesamtertrag in einem einzelnen Versuch zu erhalten, führen kann. Im Gegensatz zu dieser intrinsischen Stochastizität, bezieht sich die Unsicherheit auf die Stochastizität bezüglich der Wahl eines MDP aus vielen MDP, also die sogenannte extrinsische Stochastizität [60]. Dies entspricht dem Unterschied zwischen Standardabweichung der Messungen und absoluter Unsicherheit der Messwerte bei mehreren Stichproben.

Im Folgenden arbeiten wir mit mehrdimensionalen Objekten und deren Kovarianzmatrizen. Daher müssen diese Objekte zur Bestimmung der Kovarianzmatrix zunächst entsprechend vektorisiert werden. Dies kann durch eine beliebige, gegebenenfalls mehrdimensionale Nummerierung der Komponenten dieser Objekte geschehen. In der folgenden Notation nummerieren wir daher alle diskreten Zustände $(s_1, \ldots, s_{|S|})$ und Aktionen $(a_1, \ldots, a_{|A|})$, um den Zusammenhang zu den Kovarianzmatrizen herstellen zu können. So wird etwa ein mehrdimensionaler Matrixindex (i, j, k), der sich auf Zustands-Aktions-Folgezustands-Tripel bezieht, auf den Index $i|A||S| + j|S| + k$ abgebildet. Nur in Abschn. 10.1 bezieht sich die Nummerierung wieder auf die Beobachtungen selbst.

9.2.1 Anwendung der Unsicherheitspropagation auf die Bellman-Iteration

Das Konzept der Unsicherheitspropagation (siehe Abschn. 3.3.4) wenden wir auf die Bellman-Iteration an, die für diskrete MDP in

$$
\begin{aligned}
Q^m(s_i, a_j) &= \left(TQ^{m-1}\right)(s_i, a_j) && (9.1)\\
&= \sum_{k=1}^{|S|} P(s_k|s_i, a_j)\left(R(s_i, a_j, s_k) + \gamma V^{m-1}(s_k)\right) && (9.2)
\end{aligned}
$$

besteht. Für den Policy-Evaluation-Fall substituieren wir $V^m(s_k) = Q^m(s_k, \pi(s_k))$ und für den Policy-Iteration-Fall $V^m(s_k) = \max_{a' \in A} Q^m(s_k, a')$, wobei π die zu evaluierende Policy

ist. Im allgemeineren stochastischen Fall erfolgt die Substitution

$$V^m(s_k) = \sum_{l=1}^{|A|} \pi(s_k, a_l) Q^m(s_k, a_l). \tag{9.3}$$

Um nun die Unsicherheit der angestrebten Q-Funktionen zu bestimmen, erfolgt die Unsicherheitspropagation parallel zur Bellman-Iteration. Diese wird dazu als Fixpunkt-Iteration einer Funktion aufgefasst, deren Argumente die Modellparameter P und R sowie die Q-Funktion Q selbst sind. Wir benötigen, wie in Abschn. 3.3.4 beschrieben, dazu das erste (quadrierte) Taylor-Glied dieser Funktion für jede Iteration, das aus der Jacobi-Matrix und der Kovarianzmatrix der Argumente zusammengesetzt wird.

Für gegebene Kovarianzmatrizen $\mathbf{Cov}(P, P)$, $\mathbf{Cov}(R, R)$ und $\mathbf{Cov}(P, R)$ für die Transitionswahrscheinlichkeiten und die Rewards erhalten wir also eine vollständige initiale Kovarianzmatrix

$$\mathbf{Cov}\left(\left(Q^0, P, R\right), \left(Q^0, P, R\right)\right) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathbf{Cov}(P, P) & \mathbf{Cov}(P, R) \\ 0 & \mathbf{Cov}(P, R)^T & \mathbf{Cov}(R, R) \end{pmatrix} \tag{9.4}$$

und die Kovarianzmatrix nach der $m$ten Bellman-Iteration als

$$\mathbf{Cov}((Q^m, P, R), (Q^m, P, R)) = D^{m-1}\mathbf{Cov}((Q^{m-1}, P, R), (Q^{m-1}, P, R))(D^{m-1})^T \tag{9.5}$$

Dabei bezeichnet D^m die Jacobi-Matrix der Funktion

$$\begin{pmatrix} Q^m \\ P \\ R \end{pmatrix} = f \begin{pmatrix} Q^{m-1} \\ P \\ R \end{pmatrix} \tag{9.6}$$

$$= \begin{pmatrix} TQ^{m-1} \\ P \\ R \end{pmatrix}, \tag{9.7}$$

die sich zu

$$D^m = \begin{pmatrix} D^m_{Q,Q} & D^m_{Q,P} & D^m_{Q,R} \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix} \tag{9.8}$$

$$(D^m_{Q,Q})_{(i,j),(k,l)} = \gamma\pi(s_k, a_l)P(s_k|s_i, a_j) \tag{9.9}$$

$$(D^m_{Q,P})_{(i,j),(l,n,k)} = \delta_{il}\delta_{jn}\left(R(s_i, a_j, s_k) + \gamma V^m(s_k)\right) \tag{9.10}$$

$$(D^m_{Q,R})_{(i,j),(l,n,k)} = \delta_{il}\delta_{jn}P(s_k|s_i, a_j). \tag{9.11}$$

ergibt. Die erweiterte Signatur ist erforderlich, damit auch die Kovarianzen zwischen Q und P sowie zwischen Q und R weitergegeben werden können. Schließlich erfolgt die Unsicherheitspropagation rekursiv, wobei in jedem Rekursionsschritt die Messwerte P und R daher sowohl als Argument als auch als Wert für die Unsicherheitspropagation berücksichtigt werden müssen. Die Index-Notation erfolgt hier hierarchisch, so dass die tatsächliche mehrdimensionale Struktur in eine zweidimensionale, also in eine Matrix überführt wird. In $(D^m_{Q,Q})_{(i,j),(k,l)}$ bezeichnen etwa i und k die Nummern der Zustände sowie j und l die der Aktionen. Bezogen auf P und R stehen die jeweils drei gruppierten Indizes für die Nummern von Zustand, Aktion und Folgezustand.

Darüber hinaus sind alle Parameter von Q^m linear in Q^m, insgesamt ist Q^m eine bilineare Funktion in ihren Parametern. Die Unsicherheitspropagation kann daher näherungsweise zur Anwendung kommen [19]. Im vorliegenden Fall würde bereits die Berücksichtigung des zweiten Taylor-Gliedes die exakte Anwendung der Unsicherheitspropagation ermöglichen, jedoch auch einen erheblichen Mehraufwand erfordern.

Bei der vorgestellten Technik führt die parallele Anwendung von Bellman-Iteration und Unsicherheitspropagation dann tatsächlich zu einer eindeutigen Lösung, wie das folgende Theorem zeigt.

Satz 7. *[102] Seien $M = (S, A, P, R)$ ein endlicher MDP und der Diskontierungsfaktor $0 < \gamma < 1$ gegeben sowie C^0 eine beliebige symmetrische und positiv definite initiale Kovarianzmatrix. Dann hat die Funktion*

$$(Q^m, C^m) \quad = \quad \left(TQ^{m-1}, D^{m-1}C^{m-1}(D^{m-1})^T\right) \tag{9.12}$$

fast sicher einen eindeutigen Fixpunkt (Q^*, C^*), der unabhängig von der initialen Wahl von Q sowohl für Policy-Evaluation als auch für Policy-Iteration ist.

Beweis: Da bereits bewiesen wurde, dass $Q^m = TQ^{m-1}$ gegen einen eindeutigen Fixpunkt Q^* konvergiert [107] und Q^m in keiner Iteration von der Kovarianzmatrix C^k oder der Jacobi-Matrix D^k mit $k < m$ abhängt, bleibt noch zu zeigen, dass auch C^* eindeutig aus der Fixpunktiteration hervorgeht. Nach der $m$ten Iteration erhalten wir

$$C^m \quad = \quad \prod_{i=0}^{m-1} D^i C^0 \prod_{i=0}^{m-1} (D^i)^T. \tag{9.13}$$

Parallel zur Konvergenz von Q^m gegen Q^* konvergiert auch D^m gegen D^*. Wenn wir nun C_{conv} als die Kovarianzmatrix bezeichnen, die nach Konvergenz von Q^m berechnet wurde, dann ergibt sich schließlich

$$C^* \quad = \quad \prod_{i=0}^{\infty} D^* C_{\text{conv}} \prod_{i=0}^{\infty} (D^*)^T \tag{9.14}$$

$$\quad = \quad (D^*)^\infty C_{\text{conv}} ((D^*)^\infty)^T. \tag{9.15}$$

Durch sukzessive Matrix-Multiplikationen erhalten wir

$$(D^*)^n \quad = \quad \begin{pmatrix} (D^*)^n_{Q,Q} & \sum_{i=0}^n (D^*)^i_{Q,Q}(D^*)_{Q,P} & \sum_{i=0}^n (D^*)^i_{Q,Q}(D^*)_{Q,R} \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix}, \tag{9.16}$$

was schließlich zu

$$(D^*)^\infty \quad = \quad \begin{pmatrix} (D^*)^\infty_{Q,Q} & \sum_{i=0}^\infty (D^*)^i_{Q,Q}(D^*)_{Q,P} & \sum_{i=0}^\infty (D^*)^i_{Q,Q}(D^*)_{Q,R} \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix} \tag{9.17}$$

$$\quad = \quad \begin{pmatrix} 0 & (\mathbf{I} - (D^*)_{Q,Q})^{-1}(D^*)_{Q,P} & (\mathbf{I} - (D^*)_{Q,Q})^{-1}(D^*)_{Q,R} \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix} \tag{9.18}$$

führt, da alle Eigenwerte von $(D^*)_{Q,Q}$ kleiner als 1 sind und $\mathbf{I} - (D^*)_{Q,Q}$ für fast alle $(D^*)_{Q,Q}$ invertierbar ist. Daher existiert $(D^*)^\infty$ fast sicher, was die Existenz von C^* impliziert. Mit weiteren einfachen Matrixoperationen erhalten wir schließlich

$$C^*_{Q,Q} \quad = \quad (\mathbf{I} - (D^*)_{Q,Q})^{-1} \begin{pmatrix} (D^*)_{Q,P} & (D^*)_{Q,R} \end{pmatrix} \begin{pmatrix} \mathbf{Cov}(P,P) & \mathbf{Cov}(P,R) \\ \mathbf{Cov}(P,R)^T & \mathbf{Cov}(R,R) \end{pmatrix} \begin{pmatrix} (D^*)^T_{Q,P} \\ (D^*)^T_{Q,R} \end{pmatrix} \left((\mathbf{I} - (D^*)_{Q,Q})^{-1}\right)^T. \tag{9.19}$$

Der Fixpunkt C^* hängt also ausschließlich von den initialen Kovarianzen $\mathbf{Cov}(P,P)$, $\mathbf{Cov}(R,R)$ und $\mathbf{Cov}(P,R)$, nicht jedoch von $\mathbf{Cov}(Q,Q)$, $\mathbf{Cov}(Q,P)$ oder $\mathbf{Cov}(Q,R)$ aus C_{conv} ab und ist daher unabhängig von den Operationen, die zur Konvergenz von Q^m führen.

□

9.3 Einsatz statistischer Paradigmen – Wahl der initialen Kovarianzmatrix

Die initiale Kovarianzmatrix

$$\mathbf{Cov}((P,R),(P,R)) \quad = \quad \begin{pmatrix} \mathbf{Cov}(P,P) & \mathbf{Cov}(P,R) \\ \mathbf{Cov}(P,R)^T & \mathbf{Cov}(R,R) \end{pmatrix} \tag{9.20}$$

muss so gewählt werden, dass sie a priori Annahmen entspricht. Vorwissen über die Gestalt der Transitionswahrscheinlichkeiten und der Rewards können und sollten hier berücksichtigt werden. Wenn kein spezielles Vorwissen zur Verfügung steht, kann die Wahl auf einen sogenannten uninformierten Prior fallen, der hinreichend motiviert werden kann und einer geeigneten Regularisierung entspricht.

Wir gehen zunächst vereinfachend davon aus, dass von verschiedenen Zustands-Aktions-Paaren ausgehende Transitionen voneinander unabhängig sind. Dies ist keineswegs selbstverständlich, denn der Prozess, der die zu Grunde liegenden MDP generiert, kann beliebigen Verteilungen folgen. Insbesondere bei der Anwendung von Funktionsapproximatoren und der Fitted-Value-Iteration, auf die wir in Abschn. 10.1 eingehen werden, wird diese Fragestellung relevant, da dann ein Ähnlichkeitsmaß im Zustands-Aktions-Raum und damit auch Korrelationen zwischen benachbarten Zustands-Aktions-Paaren eine Rolle spielen können. Unter der vereinfachenden Voraussetzung jedoch können wir die Transitionswahrscheinlichkeiten, von einem Zustands-Aktions-Paar ausgehend, als Parameter einer Multinomial-Verteilung modellieren.

9.3.1 Der frequentistische Ansatz

Im frequentistischen Ansatz modellieren wir $P(s'|s,a)$ durch die relative Häufigkeit der beobachteten Transitionen und setzen

$$P(s_k|s_i,a_j) \quad = \quad \frac{n_{s_k|s_i,a_j}}{n_{s_i,a_j}}. \tag{9.21}$$

Die Kovarianzen für alle von einem Zustands-Aktions-Paar ausgehenden Transitionen, die dann einer Multinomial-Verteilung unterliegen, ergeben sich zu

$$(\mathbf{Cov}(P,P))_{(i,j,k),(l,m,n)} \quad = \quad \delta_{i,l}\delta_{j,m}\frac{P(s_k|s_i,a_j)(\delta_{k,n}-P(s_n|s_i,a_j))}{n_{s_i,a_j}-1}. \tag{9.22}$$

Eine wesentliche Vereinfachung stellt die Poisson-Näherung

$$(\mathbf{Cov}(P,P))_{(i,j,k),(l,m,n)} \quad = \quad \delta_{i,l}\delta_{j,m}\delta_{k,n}\frac{P(s_k|s_i,a_j)}{n_{s_i,a_j}-1} \tag{9.23}$$

dar, da $\mathbf{Cov}(P(\cdot|s_i,a_j),P(\cdot|s_i,a_j))$ dadurch zu einer Diagonalmatrix wird. Diese Näherung kann etwa für große Zustandsräume interessant sein, da die Unsicherheitspropagation wesentlich schneller ablaufen kann. Sie wird aber im weiteren Verlauf der Arbeit und für die Experimente in Kap. 11 nicht benutzt.

Zur Modellierung der Rewards gehen wir darüber hinaus davon aus, dass selbst die Rewards verschiedener Transitionen, die vom gleichen Zustands-Aktions-Paar ausgehen, statistisch voneinander unabhängig sind. Die frequentistische Näherung besteht dann in dem durchschnittlich beobachteten Reward

$$R(s_i,a_j,s_k) \quad = \quad \bar{r}_{s_i,a_j,s_k}, \tag{9.24}$$

wodurch $\mathbf{Cov}(R,R)$ zu einer Diagonalmatrix wird und die Form

$$\mathbf{Cov}(R(s_i,a_j,s_k),R(s_i,a_j,s_k)) \quad = \quad \frac{\mathbf{var}(r_{s_i,a_j,s_k})}{n_{s_k|s_i,a_j}-1} \tag{9.25}$$

annimmt.

9.3.2 Der Bayes'sche Ansatz

Der frequentistische Ansatz und insbesondere die Poisson-Näherung führen nur zu Näherungen der tatsächlichen Unsicherheiten. Zwar sind die frequentistischen Schätzer erwartungstreu, jedoch bei einer sehr geringen Anzahl an Beobachtungen in der Praxis ungeeignet. Im Bayes'schen Paradigma geht man daher von a priori Verteilungen [19, 59] über den Parameterraum $P(s_k|s_i, a_j)$ aus, die sich durch konkrete Beobachtungen zu a posteriori Verteilungen spezialisieren. Die Dirichlet-Verteilung mit der Dichtefunktion

$$\begin{aligned} &P(P(s_1|s_i,a_j),\ldots,P(s_{|S|}|s_i,a_j))_{\alpha_{1,i,j},\ldots,\alpha_{|S|,i,j}} \\ &\quad = \frac{\Gamma(\alpha_{i,j})}{\prod_{k=1}^{|S|}\Gamma(\alpha_{k,i,j})}\prod_{k=1}^{|S|}P(s_k|s_i,a_j)^{\alpha_{k,i,j}-1} \end{aligned} \tag{9.26}$$

und $\alpha_{i,j} = \sum_{k=1}^{|S|}\alpha_{k,i,j}$ ist ein konjugierter Prior für die Multinomial-Verteilung mit den a posteriori Parametern

$$\alpha^d_{k,i,j} \quad = \quad \alpha_{k,i,j} + n_{s_k|s_i,a_j} \tag{9.27}$$

im Lichte von $n_{s_k|s_i,a_j}$ beobachteten Transitionen von s_i nach s_k unter Anwendung der

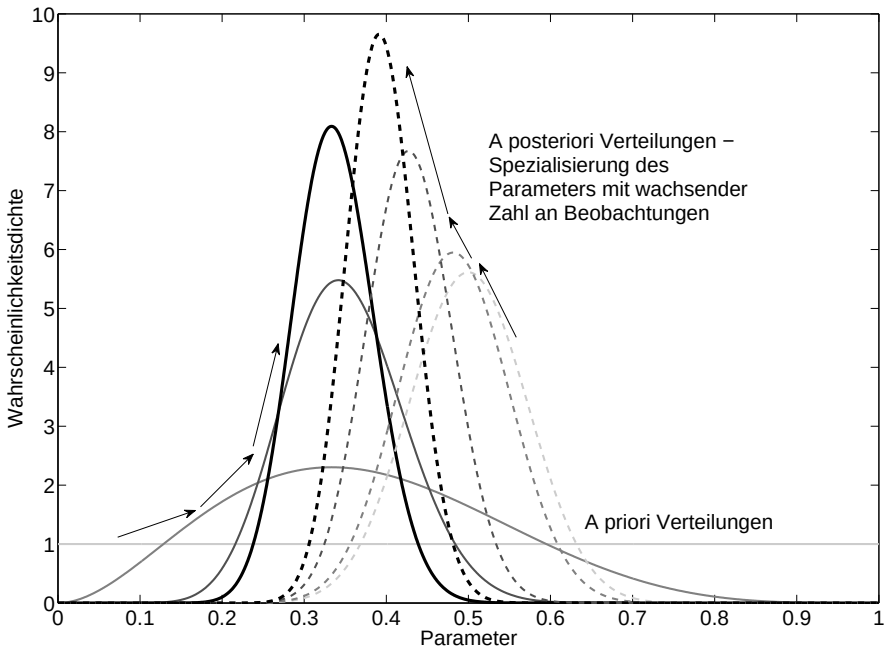


Abbildung 9.1: Illustration der Spezialisierung zweier verschiedener a priori Verteilungen eines β-verteilten Parameters einer Binomialverteilung zu a posteriori Verteilungen nach einer wachsenden Zahl an Beobachtungen. Während einer der Prior (durchgezogene Linie) uninformiert mit maximaler Entropie, also eine uniforme Verteilung ist, geht das zweite Beispiel (gestrichelte Linie) von einer starken Tendenz für den Parameter 1/2 aus. Jede der a posteriori Verteilungen ist eine Dirichlet-Verteilung. Die Posteriors sind jeweils im gleichen Linienstil gezeichnet. Der wahre Parameter ist 1/3.

Aktion a_j. Die initiale Kovarianzmatrix zur Beschreibung der Unsicherheiten der Transi-

tionswahrscheinlichkeiten erhält dann die Form

$$(\mathbf{Cov}(P,P))_{(i,j,k),(l,m,n)} = \delta_{i,l}\delta_{j,m}\frac{\alpha^d_{k,i,j}(\delta_{k,n}\alpha^d_{i,j} - \alpha^d_{n,i,j})}{(\alpha^d_{i,j})^2(\alpha^d_{i,j}+1)} \tag{9.28}$$

und geht von dem Schätzer

$$P(s_k|s_i,a_j) = \frac{\alpha^d_{k,i,j}}{\alpha^d_{i,j}} \tag{9.29}$$

für die Transitionswahrscheinlichkeiten aus.

In sehr ähnlicher Weise können die Rewards etwa als normal-verteilt angenommen werden. Dann ist dazu die Normal-Gamma-Verteilung ein konjugierter Prior. Zur uniformen Verteilung und zur Exponentialverteilung sind dagegen die Pareto- und die Gamma-Verteilung konjugierte Prior.

Letztlich sind die konjugierten Prior in beiden Fällen aber meistens ebenfalls nur Näherungen, die zu einer analytischen und meist schnellen Berechnung der a posteriori Verteilungen führt. Entscheidend bei der Wahl des Priors ist jedoch vor Allem, was man als Anwender der Methode tatsächlich zu Wissen glaubt, bevor man irgendetwas darüber sicher weiß und durch Beobachtungen belegen kann. Denn darauf beruht letztlich die Bayes'sche Philosophie.

9.4 Unsicherheitsberücksichtigende Bellman-Iteration

Ist nun die initiale Kovarianzmatrix $\mathbf{Cov}((P,R),(P,R))$ bekannt und wird zusammen mit der Bellman-Iteration und für die Unsicherheitspropagation eingesetzt, dann ergibt sich die Unsicherheit der Q-Werte schließlich aus der Kovarianzmatrix $\mathbf{Cov}(Q^*,Q^*)$, die im Fixpunkt C^* als Teilmatrix enthalten ist. Die absolute Unsicherheit ist gerade

$$\sigma Q^* = \sqrt{\mathbf{diag}(\mathbf{Cov}(Q^*,Q^*))}, \tag{9.30}$$

da die Varianzen der Q-Werte auf der Diagonalen der Kovarianzmatrix liegen.

Schließlich liefert die unsicherheitsberücksichtigende Q-Funktion

$$Q_u(s_i,a_j) = (Q^* - \xi\sigma Q^*)(s_i,a_j) \tag{9.31}$$

die garantierte erwartete Performance (siehe Abb. 9.2), also den erwarteten Gesamtertrag, mit einer gewissen Wahrscheinlichkeit an den Zuständen s_i unter Anwendung der Aktionen a_j und unter den a posteriori Annahmen, die sich aus den a priori Annahmen und den Beobachtungen ergeben, falls im Anschluss der Policy $\pi(s_k) = \max_{a'} Q^*(s_k,a')$ gefolgt wird. Der hier eingeführte Parameter $\xi \in \mathbb{R}$ definiert das Konfidenzinterval und beschreibt das Perzentil, das wiederum die Wahrscheinlichkeit determiniert, mit der die Garantieaussage zutrifft.

Nehmen wir dazu exemplarisch an, dass Q normalverteilt ist, also die Q-Werte einer Normalverteilung folgen. Dann führt die Wahl von $\xi = 2$ zu einer garantierten Performance mit der Wahrscheinlichkeit $P(2) = 0.977$.

Man beachte, dass diese Informationen über die Unsicherheit der Q-Funktion zwar zu einer Verbesserung der Exploration führen können, indem etwa Zustands-Aktions-Paare mit hoher Unsicherheit exploriert werden, jedoch die garantierte Performance selbst nicht prinzipiell verbessert werden kann, indem man den Aktionen $\arg\max_a Q_u(s,a)$ folgt. Der Agent muss der erwartungswert-optimalen Policy folgen, damit die Performance und ihre Unsicherheit konsistent geschätzt werden.

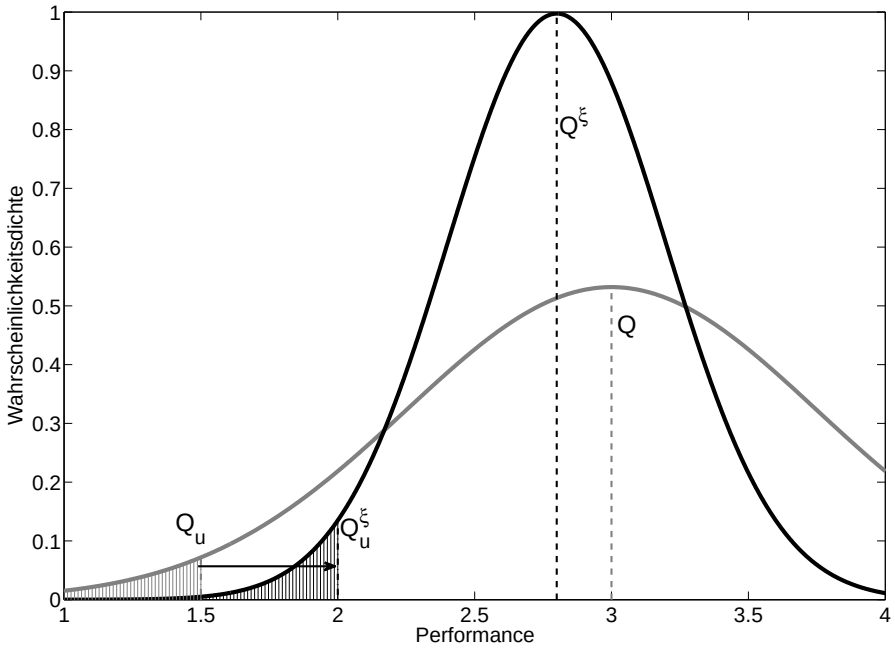


Abbildung 9.2: Illustration der Bestimmung und Maximierung der garantierten Minimalperformance. Der Wert Q_u beschreibt die Performance, die bei Anwendung von π mit einer von ξ abhängigen Wahrscheinlichkeit garantiert werden soll, die sich aus der Fläche rechts von Q_u integriert. Sicherheitsoptimalität besteht nun darin, Q_u so weit wie möglich nach rechts zu schieben, so dass die aufintegrierte Fläche gleich groß bleibt. Die daraus resultierende Q-Funktion hängt vom gewählten Perzentil ab und wird als Q^ξ geschrieben. Die Minimalperformance ist dann Q_u^ξ.

9.5 Sicherheitsoptimalität

Um dieses Problem nun zu beheben und die garantierte Performance tatsächlich zu verbessern, streben wir die sogenannte sicherheitsoptimale Policy an, die die garantierte Performance an einem zuvor spezifizierten Konfidenzlevel maximiert (siehe Abb. 9.2). Die Idee besteht darin, eine Policy π zu finden, so dass Z maximal und

$$\forall s, a : P\left(\bar{Q}^\pi(s,a) > Z(s,a)\right) \quad > \quad P(\xi) \tag{9.32}$$

erfüllt sind, wobei $\bar{Q}^\pi$ die wahre Performance-Funktion bzgl. π beschreibt und $P(\xi)$ eine zuvor festgelegte Wahrscheinlichkeit darstellt. Wir streben eine solche Lösung an, indem wir Z durch Q_u^π approximieren und

$$\pi^\xi = \arg\max_\pi \max_a Q_u^\pi(\cdot, a) \tag{9.33}$$

$$= \arg\max_\pi \max_a \left(Q^\pi - \xi\sigma Q^\pi\right)(\cdot, a) \tag{9.34}$$

unter der Nebenbedingung, dass $Q^{\pi^\xi} = Q^\xi$ die gültige Q-Funktion für π^ξ ist, also

$$\forall i, j : Q^\xi(s_i, a_j) = \sum_{k=1}^{|S|} P(s_k|s_i, a_j)\left(R(s_i, a_j, s_k) + \gamma Q^\xi(s_k, \pi^\xi(s_k))\right), \tag{9.35}$$

gilt, bestimmen. Auf die Bellman-Iteration bezogen, soll Q also ein Fixpunkt sein nicht bzgl. der Bewertungsfunktion als Maximum über alle möglichen Q-Werte, sondern als Maximum

über die Q-Werte abzüglich ihrer gewichteten Unsicherheit. Also wählen wir

$$\pi^m(s) \quad = \quad \arg\max_{a'} \left((Q^m - \xi\sigma Q^m)(s, a')\right) \tag{9.36}$$

in jeder Iteration, zusammen mit einer Aktualisierung der Unsicherheiten bzgl. π^m durch Anwendung der Unsicherheitspropagation. Auf diese Weise erhalten wir eine aus Bellman-Iteration und Unsicherheitspropagation verschränkte Iteration, die in Alg. 8 dargestellt wird.

9.5.1 Sicherheitsoptimalität erfordert stochastische Policies

Policy-Evaluation kann sowohl für deterministische als auch stochastische Policies erfolgen, jedoch existiert für jeden MDP stets eine optimale deterministische Policy [78]. Im Kontext der Sicherheitsoptimalität gilt dies nicht zwangsläufig. Tatsächlich gibt es einen Bias zu Gunsten des Betrages von $\xi\sigma Q(s, \pi(s))$ im Vergleich zu $\xi\sigma Q(s, a), a \neq \pi(s)$, falls π die ausgewertete Policy ist. Dies lässt sich dadurch erklären, dass $R(s, \pi(s), s')$ stärker mit $V(s') = Q(s', \pi(s'))$ korreliert ist als $R(s, a, s'), a \neq \pi(s)$, da die Bewertungsfunktion die Wahl von Aktion $\pi(s)$ für jedes weitere Auftreten des Zustandes s impliziert. Daher gibt es für die deterministische verschränkte Iteration keine Konvergenzgarantie, denn der Wechsel der Policy von π zu π' wegen $Q(s, \pi'(s)) - \xi\sigma Q(s, \pi'(s)) > Q(s, \pi(s)) - \xi\sigma Q(s, \pi(s))$ kann zu einer erhöhten Unsicherheit für π' in s führen und daher zu $Q'(s, \pi'(s)) - \xi\sigma Q'(s, \pi'(s)) < Q'(s, \pi(s)) - \xi\sigma Q'(s, \pi(s))$ für Q' in der folgenden Iteration, was eine Oszillation auslösen kann.

Algorithmus 8 Verschränkte Iteration zur Bestimmung sicherheitsoptimaler Policies für diskrete Markov-Entscheidungsprozesse

Bedingungen: gegebene Beobachtungen P und R eines diskreten MDP, initiale Kovarianzmatrizen $\mathbf{Cov}(P, P)$, $\mathbf{Cov}(R, R)$, and $\mathbf{Cov}(P, R)$ sowie Skalare $0 < \gamma < 1$ und ξ

Zusicherungen: bestimmt die sicherheitsoptimale Q-Funktion Q, ihre Unsicherheit σQ und Policy π

setze $C^0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathbf{Cov}(P, P) & \mathbf{Cov}(P, R) \\ 0 & \mathbf{Cov}(P, R)^T & \mathbf{Cov}(R, R) \end{pmatrix}$

setze $\forall i, j : Q^0(s_i, a_j) = 0$

setze $\forall i, j : \pi^0(s_i, a_j) = \frac{1}{|A|}$

setze $t = 0$

solange die gewünschte Genauigkeit nicht erreicht ist **führe aus**

setze $t = t + 1$

setze $\forall i, j : (\sigma Q^{t-1})(s_i, a_j) = \sqrt{(C^{t-1})_{i|A|+j, i|A|+j}}$

finde $\forall i : a_{i,\max} = \arg\max_a (Q^{t-1} - \xi\sigma Q^{t-1})(s_i, a)$

setze $\forall i : d_i = \min\left(\frac{1}{t}, 1 - \pi^{t-1}(s_i, a_{i,\max})\right)$

setze $\forall i : \pi^t(s_i, a_{i,\max}) = \pi^{t-1}(s_i, a_{i,\max}) + d_i$

setze $\forall i : \forall a_j \neq a_{i,\max} : \pi^t(s_i, a_j) = \frac{1 - \pi^t(s_i, a_{i,\max})}{1 - \pi^t(s_i, a_{i,\max}) + d_i} \pi^{t-1}(s_i, a_j)$

setze $\forall i, j : Q^t(s_i, a_j) = \sum_{k=1}^{|S|} P(s_k | s_i, a_j) \left(R(s_i, a_j, s_k) + \gamma \sum_{l=1}^{|A|} \pi^t(s_k, a_l) Q^{t-1}(s_k, a_l) \right)$

setze $D = \begin{pmatrix} D_{Q^t, Q^{t-1}} & D_{Q^t, P} & D_{Q^t, R} \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix}$

setze $C^t = D C^{t-1} D^T$

ende solange

gebe Q^t, σQ^t und π^t zurück

Es ist intuitiv einleuchtend, dass eine sicherheitsoptimale Policy im Allgemeinen stochastisch ist, da die Gewinne an Performance und Sicherheit ausgeglichen werden müssen. Das

Risiko, einen niedrigen erwarteten Gesamtertrag zu erhalten, wird schließlich auch dadurch reduziert, dass man die Policy über eine Menge geeigneter Aktionen streut. Dies entspricht der aus der Finanzwirtschaft bekannten Strategie der Diversifikation (siehe Abb. 9.3).

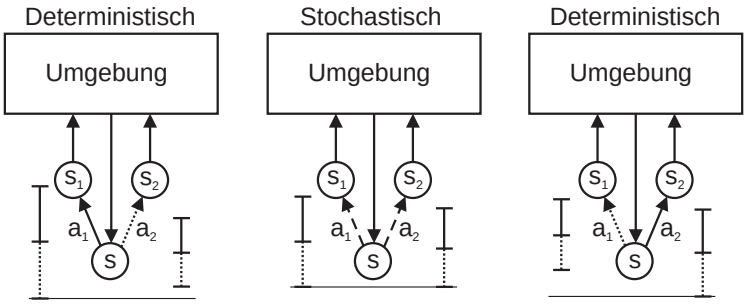


Abbildung 9.3: Illustration des Prinzips der Diversifikation. Die Q-Werte von Aktionen, die zur aktuellen Policy gehören, haben in der Tendenz eine größere Unsicherheit, da alle anderen Aktionen nur einmal ausgeführt würden. Daher kann der Wechsel der Policy zu einem Ansteigen der Perzentil-Performance für die betroffene Aktion kommen, wodurch diese wieder bevorzugt werden kann. Die Lösung dieses Problems liegt in der Diversifikation. Sicherheitsoptimale Policies sind im Allgemeinen stochastisch, denn nur so können Performance und Unsicherheit zwischen den verschiedenen Aktionen abgewogen werden.

Der Wert ξ entscheidet letztlich über die Kosten der Sicherheit. Wenn $\xi > 0$ groß ist, dann wird die sicherheitsoptimale Policy eher stochastisch und man zahlt einen Preis für den Gewinn an Sicherheit einer niedrigeren Performance, während ein kleines $\xi \leq 0$ deterministische sicherheitsoptimale Policies garantiert. Hier erhält Unsicherheit die Bedeutung von Chance auf eine hohe Performance.

Aus diesem Grund erweitern wir die verschränkte Iteration und definieren schließlich die stochastische sicherheitsoptimale Bellman-Iteration als

$$\begin{pmatrix} Q^m \\ C^m \\ \pi^m \end{pmatrix} = \begin{pmatrix} TQ^{m-1} \\ D_{m-1}C^{m-1}D_{m-1}^T \\ \Lambda(\pi^{m-1}, TQ^{m-1}, m) \end{pmatrix} \tag{9.37}$$

mit

$$\Lambda(\pi, Q, t)(s,a) = \begin{cases} \pi(s,a) + \min\left(\frac{1}{t}, 1-\pi(s,a)\right) & : \quad a = a_Q(s) \\ \frac{1-\min\left(\pi(s,a_Q(s))+\frac{1}{t},1\right)}{1-\pi(s,a_Q(s))}\pi(s,a) & : \quad \text{sonst} \end{cases} \tag{9.38}$$

und

$$a_Q(s) = \arg\max_{a'} \left((Q - \xi\sigma Q)(s,a')\right). \tag{9.39}$$

Die harmonisch reduzierte Rate zur Veränderung der Aktionsauswahlwahrscheinlichkeiten garantiert per Konstruktion sowohl die Erreichbarkeit jeder stochastischen Policy als auch Konvergenz (siehe auch Abschn. 6.4). In Alg. 8 wird die Funktionsweise dieses Verfahrens zusammengefasst.

Die Funktion

$$Q_u^\xi(s_i, a_j) = \left(Q^\xi - \xi\sigma Q^\xi\right)(s_i, a_j) \tag{9.40}$$

mit (Q^ξ, C^ξ, π^ξ) als dem Fixpunkt der stochastischen verschränkten Bellman-Iteration bei gegebenem $\xi \in \mathbb{R}$ liefert die garantierte erwartete Performance, also den erwarteten Gesamtertrag, mit einer gewissen Wahrscheinlichkeit, an den Zuständen s_i unter Anwendung der Aktionen a_j und unter den a posteriori Annahmen, die sich aus den a priori Annahmen und den Beobachtungen ergeben, falls im Anschluss der stochastischen Policy π^ξ gefolgt wird. Insbesondere ist diese Performance für die spezielle Wahl von ξ maximal.

Die Zeitkomplexität des Algorithmus ist natürlich höher als die der Standard-Bellman-Iteration, die für diskrete MDP $O(|S|^2|A|)$ beträgt ($O(|S|^2|A|^2)$ für stochastische Policies). Der aufwändigste Prozess besteht im Aktualisieren der Kovarianz-Matrix, was einen Zeitaufwand zwischen $\Omega((|S||A|)^2 \log((|S||A|)))$ und $O((|S||A|)^{2.376})$ erfordert [15], da jeder Eintrag von Q von höchstens $|S|$ Einträgen in P und R abhängen. Damit bleibt jedoch die Zeitkomplexität insgesamt durch diese Größenordnungen beschränkt. Betrachtet man die Anzahl der Aktionen als durch eine Konstante beschränkt, dann ist das hier beschriebene Verfahren mindestens um $\Omega(\log(|S|))$, höchstens jedoch um $O(|S|^{0.376})$ langsamer als die Standard-Bellman-Iteration.

9.6 Anwendungen

Die hier vorgestellten Verfahren sind geeignet, eine Reihe möglicher Anwendungsfelder zu erschließen, die in der Praxis, besonders im industriellen Umfeld, von großer Bedeutung sind.

9.6.1 Qualitätssicherung durch Unsicherheitspropagation

Mit einem positiven ξ ermittelt man eine garantierte Minimalperformance für eine gegebene oder die optimale Policy. Das Konzept der Sicherheitsoptimalität ermöglicht darüber hinaus, diese Minimalperformance zu optimieren. Die wesentliche Motivation dafür besteht darin, keine minderwertige Policy auszuliefern und das Risiko, eine ungeeignete Policy zu erhalten, trotz des Anscheins einer guten Performance, zu verringern. Und gerade den Zugriff auf eine konkrete Quantisierung der Unsicherheit zu gewährleisten, ermöglicht ein Urteil über die Vertrauenswürdigkeit des Ergebnisses. Ist die garantierte Performance an einem spezifizierten Startzustand unzureichend, müssen weitere Beobachtungen gesammelt werden.

Falls die Exploration teuer ist, das zu regelnde System jedoch sicherheitskritisch, so dass eine gewisse Performance-Wahrscheinlichkeit definitiv erfüllt sein muss, dann ist es sinnvoll, sich das Konzept der Sicherheitsoptimalität zu Nutze zu machen. Man gibt die Optimalität des Erwartungswertes zu Gunsten der Optimalität an einem spezifizierten Perzentil auf.

9.6.2 Wettkämpfe und Exploration

Symmetrisch dazu, für negative ξ, nutzt man die Unsicherheit in entgegengesetzter Weise durch Ausnutzen der Chance auf eine hohe Performance. Dies ist von besonderem Interesse für eine zielgerichtete Exploration, so dass verstärkt Zustands-Aktions-Paare exploriert werden, für die $Q_u^\xi(s,a)$ groß ist. Dort ist der Schätzer für den entsprechenden Q-Wert bereits groß, aber die wahre Performance könnte noch wesentlich besser sein, wenn zusätzlich noch die Unsicherheit entsprechend groß ist. Für eine zielgerichtete und informationssuchende Exploration kann das Verfahren also alternativ oder in Kombination mit Explorationsstrategien in sicherheitskritischen Umgebungen [38, 39] verwendet werden.

Ein weiteres Anwendungsfeld besteht in Wettkampfsituationen, die sich ebenfalls symmetrisch zur Qualitätssicherung beschreiben lassen. Der Agent soll hier einer Policy folgen, die ihm die Chance bietet, außerordentlich gut abzuschneiden und schließlich den Wettkampf zu gewinnen. Auch in diesem Fall ist das Konzept der Sicherheitsoptimalität von Bedeutung, da nicht die erwartete Performance, sondern die Perzentil-Performance das entscheidende Kriterium ist und maximiert werden soll.

9.6.3 Steigerung der Informationseffizienz bei praktischen Anwendungen

Neben der Qualitätssicherung besteht das zweite Hauptziel der hier vorgestellten Verfahren in der Verbesserung der Informationseffizienz. Das Berücksichtigen von Unsicherheiten kann im Reinforcement-Learning selbst die erwartete Performance und die konkreter MDP in vielen praktischen und industriellen Anwendungen verbessern, bei denen Exploration teuer

und nur in begrenztem Umfang um vorgesehene Arbeitspunkte möglich ist. Die verfügbare Datenmenge ist in solchen Fällen gering und die Exploration findet in einer, zum Teil extrem asymmetrischen Weise statt, so dass Daten vor allem in Zustandsbereichen gesammelt werden, für die in dem technischen System bereits bevorzugt operiert wird.

Viele dieser unzureichend explorierten sogenannten Randzustände sind zwar unerwünscht im Erwartungswert, aber nicht zwangsläufig im Einzelfall (Randphänomen). Wenn der Rand hinreichend groß ist, dann geschieht es wenigstens einige Male, dass einer der Ausreißer einen hohen Reward vortäuscht. Dies kann dazu führen, dass die vermeintlich optimale Policy solche Zustände anstrebt und sich dadurch zu weit von den Arbeitspunkten entfernt, was eine geringere Performance zur Folge haben würde. Daher führt ein umsichtiges Berücksichtigen der Unsicherheit dazu, dass der Agent solche Ausreißer in seiner Policy unberücksichtigt lässt und Aktionen ausführt, die gesichert eine hohe Performance zur Folge haben.

Man beachte dabei, dass die Verbesserung des Erwartungswertes durch die Bestimmung sicherheitsoptimaler Policies keinen Widerspruch zu den Zielstellungen des klassischen Verfahrens verursacht, das ja bereits erwartungswert-optimal arbeitet. Das Phänomen kommt durch die asymmetrische Exploration zu Stande, die zu ungeeigneten a priori Annahmen über die Parameter der Umgebung führt und ist daher nicht alleine auf Eigenschaften des MDP zurückzuführen. Es ist ferner zu beachten, dass die Größe der Randregion mit der Dimensionalität des Zustandsraumes ansteigt und die Berücksichtigung dieser Problematik daher bei hochdimensionalen Problemen von großer Bedeutung ist.

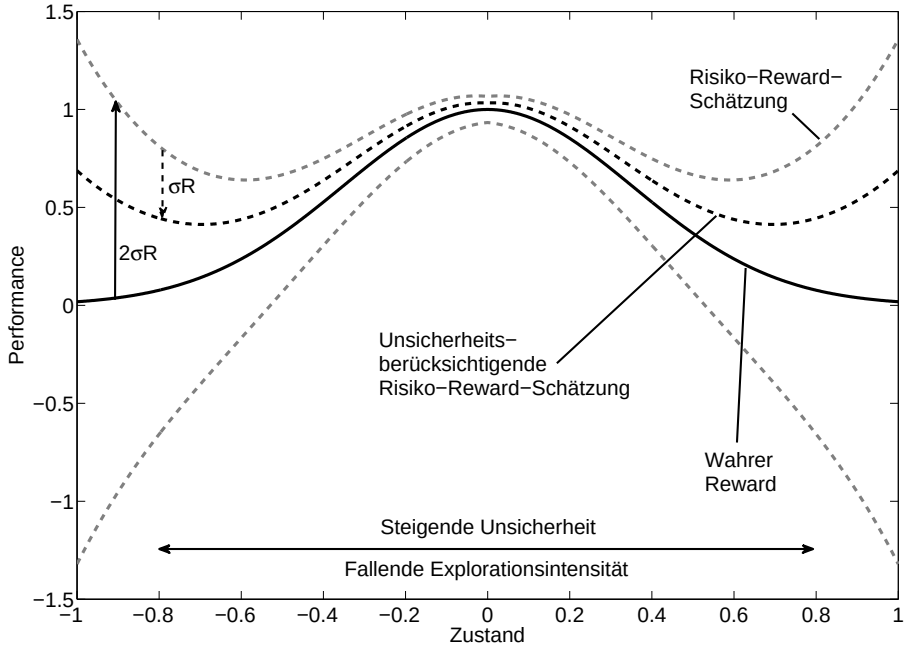


Abbildung 9.4: Illustration des Randphänomens. Bei Problemklassen, die in der Nähe von Arbeitspunkten tendenziell eine höhere Performance erzielen als in einer größeren Entfernung zu ihnen (schwarze durchgezogene Linie) und für die sich der Agent während der Exploration vorwiegend in der Nähe dieser Arbeitspunkte aufhält, kommt das sogenannte Randphänomen zum Tragen. Durch die schwach ausgeprägte Exploration am Rand kommt es dort zu einer erhöhten Unsicherheit bzgl. des Rewards und damit zu einem erhöhten Risiko (grau gestrichelte Linie), im Einzelfall, also für einzelne Zustands-Aktions-Paare fälschlich eine hohe Performance zu vermuten. Die UPRL kann dieses Risiko soweit ausgleichen (schwarze gestrichelte Linie), dass die sicherheitsoptimale Policy weiterhin dem Rand ausweicht und sich in der Nähe eines Arbeitspunktes aufhält.

Kapitel 10

Allgemeine Anwendbarkeit – Funktionsapproximatoren und Erweiterte Statistik

10.1 Anwendung auf Funktionsapproximatoren

Bisher haben wir uns ausschließlich dem diskreten Fall gewidmet, also MDP mit endlichen Zustandsräumen betrachtet. Da dies praktische Einschränkungen zur Folge haben kann, besonders bei industriellen Problemstellungen, wollen wir uns in diesem Kapitel verschiedenen möglichen Umsetzungen für Funktionsapproximatoren zusammen mit der Fitted-Value-Iteration zuwenden, wobei wir uns auf die Least-Squares-Policy-Iteration konzentrieren.

Dabei sind die unsicherheitsbehafteten Parameter nicht mehr die Transitionswahrscheinlichkeiten und Rewards, sondern die Beobachtungen selbst, also die beobachteten Folgezustände und Rewards.

10.1.1 Least-Squares-Policy-Iteration

Bei der hier betrachteten Variante der Least-Squares-Policy-Iteration (siehe Abschn. 4.1.1) gehen wir von einer endlichen Anzahl von Aktionen aus. Die Q-Funktion ist eine lineare Funktion und basiert auf Features Φ, so dass $Q_{\mathbf{w}}(s,a) = \Phi(s,a)^T\mathbf{w}$. Die Bellman-Iteration besteht dann im Wesentlichen im iterativen Lösen des linearen Gleichungssystems

$$\begin{aligned} \left(\Phi^T\Phi + \lambda\mathbf{I}\right)\mathbf{w}^m &= \Phi^T\left(R + \gamma(V')^{m-1}\right) && (10.1)\\ &= \Phi^T\left(R + \gamma(\Phi')^{m-1}\mathbf{w}^{m-1}\right) && (10.2)\end{aligned}$$

für $\mathbf{w}^m$ mit $\lambda \geq 0$ als Regularisierungsparameter, Φ der Feature-Matrix aller beobachteten Zustands-Aktions-Paare und $(\Phi')^m = \sum_{a=1}^{|A|}\left(\left(\pi^m(\cdot,a)\mathbf{1}^T\right) \bullet \Phi'_a\right)$ mit Φ'_a der Feature-Matrix aller Folgezustände zusammen mit der Aktion a, $\bullet$ beschreibt dabei die komponentenweise Multiplikation und $\mathbf{1}$ einen Spaltenvektor kompatibler Größe, der aus Einsen besteht.

Die für die Unsicherheitspropagation bedeutenden Messgrößen P und R im diskreten Fall sind nun die Beobachtungen selbst, also die Features Φ' der Folgezustände und die Rewards $R = (r_1, \ldots, r_n)^T$, wobei n die Anzahl der Beobachtungen ist. In vergleichbarer Weise erhalten wir mit

$$\begin{aligned} \mathbf{v}^m &= \left(\Phi^T\Phi + \lambda\mathbf{I}\right)^{-1}\Phi^T\left(R + \gamma(V')^m\right) && (10.3)\\ Z_a^m &= (\Phi'_a)^m\mathbf{v}^m && (10.4)\end{aligned}$$

Algorithmus 9 Verschränkte Iteration zur Bestimmung sicherheitsoptimaler Policies für die Least-Squares-Policy-Iteration

Bedingungen: gegebene Beobachtungen $((\phi(s_1), a_1, r_1, \phi(s'_1)), \ldots, (\phi(s_l), a_l, r_l, \phi(s'_l))$ eines MDP, initiale Kovarianzmatrizen $\mathbf{Cov}(\Phi', \Phi')$, $\mathbf{Cov}(R, R)$ und $\mathbf{Cov}(\Phi', R)$ sowie Skalare $0 < \gamma < 1$ und ξ

Zusicherungen: bestimmt eine sicherheitsoptimale Q-Funktion Q, ihre Unsicherheit σQ und Gewichtsvektor $\mathbf{w}$

setze $C^0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{Cov}(\Phi', \Phi') & \mathbf{Cov}(\Phi', R) \\ 0 & 0 & \mathbf{Cov}(\Phi', R)^T & \mathbf{Cov}(R, R) \end{pmatrix}$

setze $\forall i : (\mathbf{w}^0)_i = 0$

setze $\forall i, a : Q^0(s'_i, a) = 0$

setze $\forall i, a : \pi^0(s'_i, a) = \frac{1}{|A|}$

setze $t = 0$

solange die gewünschte Genauigkeit nicht erreicht ist **führe aus**

setze $t = t + 1$

setze $\forall i, a : (\sigma Q^{t-1})(s'_i, a) = \sqrt{(C^{t-1})_{\dim(\Phi)+i|A|+a, \dim(\Phi)+i|A|+a}}$

finde $\forall i : a_{i,\max} = \arg\max_a (Q^{t-1} - \xi \sigma Q^{t-1})(s'_i, a)$

setze $\forall i : d_i = \min\left(\frac{1}{t}, 1 - \pi^{t-1}(s'_i, a_{i,\max})\right)$

setze $\forall i : \pi^t(s'_i, a_{i,\max}) = \pi^{t-1}(s'_i, a_{i,\max}) + d_i$

setze $\forall i : \forall a \neq a_{i,\max} : \pi^t(s'_i, a) = \frac{1 - \pi^t(s'_i, a_{i,\max})}{1 - \pi^t(s'_i, a_{i,\max}) + d_i} \pi^{t-1}(s_i, a)$

löse $\mathbf{w}^t = \left(\Phi^T \Phi + \lambda \mathbf{I}\right)^{-1} \Phi^T \left(R + \gamma \sum_{i=1}^{|A|} \left(\pi^t(\cdot, a) \bullet Q^{t-1}(\cdot, a)\right)\right)$

setze $\forall a : Q^t(\cdot, a) = \Phi'_a \mathbf{w}^t$

setze $D = \begin{pmatrix} 0 & D_{\mathbf{w}_t, Q^{t-1}} & 0 & D_{\mathbf{w}_t, R} \\ 0 & D_{Q^t, Q^{t-1}} & D_{Q^t, \Phi'} & D_{Q^t, R} \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{pmatrix}$

setze $C^t = DC^{t-1}D^T$

ende solange

setze $\forall i : (\sigma \mathbf{w}_i^t) = \sqrt{C_i^t}$

löse $\sum_{i=1}^{n} \Phi_i^T \hat{\mathbf{w}} \rightarrow \min$ unter den Nebenbedingungen $\forall i : \mathbf{w}_i^t - \xi(\sigma \mathbf{w}_i) \leq \hat{\mathbf{w}}_i \leq \mathbf{w}_i^t + \xi(\sigma \mathbf{w}_i^t)$

gebe Q^t, σQ^t und $\hat{\mathbf{w}}$ zurück

eine erweiterte Darstellung der Bellman-Iteration

$$\begin{pmatrix} \mathbf{w}^m \\ \begin{pmatrix} (Q'(\cdot,1))^m \\ \vdots \\ (Q'(\cdot,|A|))^m \end{pmatrix} \\ \Phi' \\ R \end{pmatrix} = \begin{pmatrix} \mathbf{v}^{m-1} \\ \begin{pmatrix} Z_1^{m-1} \\ \vdots \\ Z_{|A|}^{m-1} \end{pmatrix} \\ \Phi' \\ R \end{pmatrix}. \tag{10.5}$$

Die Jacobi-Matrix ergibt sich dann als die Multiplikation der Jacobi-Matrizen der Teiloperationen, die in der Berechnung des Gewichtsvektors (Gl. 10.3) und der darauffolgenden Aktualisierung der Q-Werte der Folgezustände (Gl. 10.4) bestehen. Insgesamt erhalten wir

also

$$
\begin{aligned}
D^m &= \begin{pmatrix} \mathbf{I} & 0 & 0 & 0 \\ D^m_{Q',\mathbf{w}} & 0 & D^m_{Q',\Phi'} & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} 0 & D^m_{\mathbf{w},Q'} & 0 & D^m_{\mathbf{w},R} \\ 0 & \mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{pmatrix} && (10.6)\\
&= \begin{pmatrix} 0 & D^m_{\mathbf{w},Q'} & 0 & D^m_{\mathbf{w},R} \\ 0 & D^m_{Q',\mathbf{w}} D^m_{\mathbf{w},Q'} & D^m_{Q',\Phi'} & D^m_{Q',\mathbf{w}} D^m_{\mathbf{w},R} \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{pmatrix} && (10.7)\\
&= \begin{pmatrix} 0 & D^m_{\mathbf{w},Q'} & 0 & D^m_{\mathbf{w},R} \\ 0 & D^m_{Q',Q'} & D^m_{Q',\Phi'} & D^m_{Q',R} \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{pmatrix} && (10.8)
\end{aligned}
$$

mit den Einträgen

$$
\begin{aligned}
D^m_{\mathbf{w},(Q'(\cdot,a))} &= \gamma \left(\left(\Phi^T \Phi + \lambda \mathbf{I} \right)^{-1} \Phi^T \right) \bullet \left(\mathbf{1} \pi(\cdot, a)^T \right) && (10.9)\\
D^m_{\mathbf{w},R} &= \left(\Phi^T \Phi + \lambda \mathbf{I} \right)^{-1} \Phi^T && (10.10)\\
D^m_{Q'(\cdot,a),Q'(\cdot,b)} &= D^m_{Q'(\cdot,a),\mathbf{w}} D^m_{\mathbf{w},Q'(\cdot,b)} && (10.11)\\
&= \left(\Phi'_a \cdot \gamma \left(\left(\Phi^T \Phi + \lambda \mathbf{I} \right)^{-1} \Phi^T \right) \right) \bullet \left(\mathbf{1} \pi(\cdot, b)^T \right) && (10.12)\\
&= \gamma \left(\Phi'_a \left(\Phi^T \Phi + \lambda \mathbf{I} \right)^{-1} \Phi^T \right) \bullet \left(\mathbf{1} \pi(\cdot, b)^T \right) && (10.13)\\
D^m_{Q'(s_i,a),(\Phi'_b)_{j,\cdot}} &= \delta_{i,j} \delta_{a,b} \mathbf{w}^T && (10.14)\\
D^m_{Q'(\cdot,a),R} &= D^m_{Q'(\cdot,a),\mathbf{w}} D^m_{\mathbf{w},R} && (10.15)\\
&= \Phi'_a \left(\Phi^T \Phi + \lambda \mathbf{I} \right)^{-1} \Phi^T. && (10.16)
\end{aligned}
$$

Bei dieser Notation haben wir im Gegensatz zu den Gln. 9.9, 9.10 und 9.11 vereinfachend die Matrixschreibweise gewählt. Dazu wurden jeweils eindimensionale Unter-Parametersätze ausgewählt, nach denen oder die abgeleitet werden.

Wir operieren auf einer initialen Kovarianzmatrix der Form

$$
\begin{aligned}
&\mathbf{Cov}\left(\left(\mathbf{w}^0, (Q')^0, \Phi', R \right), \left(\mathbf{w}^0, (Q')^0, \Phi', R \right) \right) \\
&= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{Cov}(\Phi', \Phi') & \mathbf{Cov}(\Phi', R) \\ 0 & 0 & \mathbf{Cov}(\Phi', R)^T & \mathbf{Cov}(R, R) \end{pmatrix}. && (10.17)
\end{aligned}
$$

Die Unsicherheitspropagation selbst läuft wie in Abschn. 9.2 beschrieben. Man erhält ein zu Satz 7 vergleichbares Konvergenz-Ergebnis, falls $\Phi'_a \left(\Phi^T \Phi + \lambda \mathbf{I} \right)^{-1} \Phi^T$ eine Nicht-Expansion für alle Aktionen a ist. Analog zu Satz 7 führt die sukzessive Matrixmultiplikation dann zu den Kovarianzmatrizen für Q-Funktion und Gewichtsvektor $C^\infty_{Q',Q'}$ und $C^\infty_{\mathbf{w},\mathbf{w}}$, die nur von Φ' und R sowie deren Kovarianzmatrizen abhängen.

Der sicherheitsoptimale Gewichtsvektor $\mathbf{w}_u$ kann schließlich durch Lösen des linearen Programms $(\Phi \mathbf{w}_u)^T \mathbf{1} \rightarrow \min$ unter den Nebenbedingungen $\mathbf{w} - \xi\sigma\mathbf{w} \leq \mathbf{w}_u \leq \mathbf{w} + \xi\sigma\mathbf{w}$ gewonnen werden.

Ein offenes Problem ist die Darstellbarkeit sicherheitsoptimaler stochastischer Policies. Eine Möglichkeit besteht in der Betrachtung stochastischer Gewichtsvektoren. Dies ist jedoch Teil zukünftiger Arbeiten.

10.1.2 Kernel-basiertes Reinforcement-Learning und Neural-Fitted-Q-Iteration

Das kernel-basierte Reinforcement-Learning besteht in einer einfachen Regression durch Kernel-Glättung (siehe Abschn. 4.1.2). Wie bei der LSPI wird auch hier die Unsicherheit der Beobachtungen selbst propagiert. Die Jacobi-Matrix bezieht sich dabei auf die Q-Werte $Q^m(s_i, a_i)$ an den Stützstellen und hat die Struktur

$$D^m = \begin{pmatrix} D^m_{Q,Q} & D^m_{Q,S'} & D^m_{Q,R} \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{pmatrix}. \tag{10.18}$$

Die Unsicherheitspropagation bei der Neural-Fitted-Q-Iteration praktisch anzuwenden, ist mit großem Aufwand verbunden, da sich in jedem Gradientenschritt die Parameter des neuronalen Netzes verändern und dementsprechend auch die Unsicherheiten berechnet werden müssen. Der Übergang der Parameter θ^m nach θ^{m+1} erfolgt in mehreren Zwischenschritten, den Gradientenabstiegsschritten, die wir mit t indizieren. Diese bestehen in der Modifikation der Parameter gemäß Gl. 4.32. Wir betrachten, wie bei der LSPI, eine zweischrittige Unsicherheitspropagation. Zu Beginn und nach Abschluss einer Iteration der NFQ erfolgt die Bestimmung der Unsicherheit von $V^m = V(\theta^m, s'_1, \ldots, s'_l)$. Die entsprechende Jacobi-Matrix ergibt sich aus der Darstellung des neuronalen Netzes. Die Kovarianzmatrix $\mathbf{Cov}(S', S')$ muss wieder initial gefüllt werden, während $\mathbf{Cov}(\theta^0, \theta^0)$ und $\mathbf{Cov}(\theta^0, S')$ initial nicht unsicherheitsbehaftet sind.

Während des Gradientenabstiegs ist also die Unsicherheit $\mathbf{Cov}(V^m, V^m)$ anstatt $\mathbf{Cov}(S', S')$ zu berücksichtigen. Wir beschreiben jeden Gradientenschritt nach θ^m_{t+1} als Funktion, die von den Parametern des vorherigen Schrittes θ^m_t und den Werten $V^m(s'_i)$ sowie r_i abhängt.

Abgesehen vom erforderlichen Aufwand, sind neuronale Netze nicht-lineare Funktionsapproximatoren. Daher muss im Einzelfall geprüft werden, ob die Unsicherheitspropagation näherungsweise angewendet werden kann oder eine Taylor-Entwicklung höheren Grades zur Anwendung kommen muss.

10.1.3 Wahl der initialen Kovarianzmatrix

Das Grundprinzip der Bestimmung der initialen Kovarianzmatrix ist identisch zu dem im diskreten Fall. Sie muss so gefüllt werden, dass sie die Vorannahmen des Anwenders repräsentiert. In der Praxis ist diese Anforderung jedoch schwer umsetzbar, da die Unsicherheit nun nicht mehr auf P und R, sondern auf den Beobachtungen selbst definiert wird. Wir schlagen daher exemplarisch vor, eine einfache kernel-basierte Regression der Rewards und Folgezustände vorzunehmen, die zu Funktionen $f : S \to \mathbb{R}$ und $g : S \to \Phi(S)$ führen. Der Residualfehler wird dann als Maß für die Unsicherheit der entsprechenden Messgröße herangezogen. Das Vorwissen über die Unsicherheit kann dazu in den Kernel integriert werden.

Die Schätzung der Rewards beispielsweise erfolgt dann mittels

$$\tilde{R}_i = \frac{K((s_i, a_i), \cdot)R}{\sum_{j=1}^{l} K((s_i, a_i), (s_j, a_j))}, \tag{10.19}$$

so dass die Differenz zwischen R_i und $\tilde{R}_i$ als Standardabweichung an R_i herangezogen werden kann, die ebenfalls kernel-basiert als

$$\mathbf{std}\left(\tilde{R}\right)_i = \frac{K((s_i, a_i), \cdot)\left|\tilde{R} - R\right|}{\sum_{j=1}^{l} K((s_i, a_i), (s_j, a_j))} \tag{10.20}$$

geglättet wird. Als Unsicherheit erhält man schließlich

$$\sigma_{R_i} = \frac{\mathbf{std}\left(\tilde{R}\right)_i}{\sqrt{\sum_{j=1}^{l} K((s_i,a_i),(s_j,a_j))}}. \tag{10.21}$$

Dies geht jedoch von der Annahme aus, dass der Kernel abstandsbasiert ist und $\forall s,a$: $K((s,a),(s,a)) = 1$ gilt. Dann kann $\sum_{j=1}^{l} K((s_i,a_i),(s_j,a_j))$ als ein Maß für die Anzahl an Beobachtungen am Zustand s_i aufgefasst werden. Die vollständige Kovarianzmatrix kann im einfachsten Fall etwa mit

$$\mathbf{Cov}(R,R)_{i,j} = \delta_{i,j}\sigma_{R_i}^2 \tag{10.22}$$

gefüllt werden. Diese Schätzung der Unsicherheit der Rewards, die so auch auf die Folgezustände angewendet werden kann, dient nur als Vorschlag. Es bleibt dem Nutzer vorbehalten, wie er die Unsicherheiten angemessen bestimmt.

10.2 Erweiterte Statistik

Das bisher vorgestellte Verfahren ermöglicht die Bestimmung und Maximierung einer garantierten Minimalperformance, jedoch unter bestimmten Einschränkungen. Dies betrifft, abgesehen von der angenommenen Linearität der Bellman-Iteration, die Symmetrie der Verteilungen der Q-Werte und die Lokalität der betrachteten Unsicherheit. Dieser Abschnitt beschäftigt sich mit der Bewältigung der letzteren beiden Aspekte. Wir betrachten der Einfachheit halber dazu wieder diskrete Zustandsräume.

10.2.1 Nutzung der vollständigen Kovarianzmatrix

Die UPRL liefert als Ergebnis die Q-Funktion einer zu evaluierenden, der optimalen oder der sicherheitsoptimalen Policy zusammen mit einer vollständigen Kovarianzmatrix, die die Unsicherheit der Q-Funktion beschreibt. Bei der weiteren Auswertung der Ergebnisse und dem Ausführen der Policy wird jedoch nur die Diagonale $\sigma Q^* = \sqrt{\mathbf{diag}(\mathbf{Cov}(Q^*,Q^*))}$ der Kovarianzmatrix berücksichtigt, die die Unsicherheiten der einzelnen Q-Werte beschreibt. Dabei bleiben die Korrelationen zwischen verschiedenen Zuständen und Zustands-Aktions-Paaren für die Analyse der Ergebnisse unberücksichtigt.

10.2.1.1 Finale Kovarianzmatrix für weiterführende statistische Analysen

Die finale Kovarianzmatrix als Ergebnis der Unsicherheitspropagation ermöglicht weiterführende statistische Analysen. Besteht etwa eine hohe Korrelation zwischen den Q-Werten zweier Zustände, so kann dies bei einer weiteren Exploration und der Anpassung der Q-Funktion berücksichtigt werden. Weniger hoch korrelierte Zustände müssen verstärkt exploriert werden. Zustände mit hohen betragsmäßigen Korrelationen zu vielen anderen Zuständen sind repräsentativ für diese, daher besteht dort geringerer Explorationsbedarf. Durch die Beobachtung eines weiteren Rewards $r_{s,a,s'}$ kann nicht nur $Q(s,a)$ angepasst werden, sondern auch $Q(\hat{s},\hat{a})$ für große $|\mathbf{Cov}(Q(s,a),Q(\hat{s},\hat{a}))|$. Dadurch kann der Gesamtbedarf an zu explorierenden Beobachtungen minimiert werden.

Jedoch ist der Gewinn durch diese Informationen für die Bestimmung optimaler Q-Funktionen und Policies für bereits gegebene Trajektorien nicht substanziell, da zum Zeitpunkt einer neuen Beobachtung die Bellman-Iteration, auch wenn sie teuer ist, im Prinzip von Neuem durchgeführt werden kann und dann wieder die Betrachtung der Diagonalen der Kovarianzmatrix genügt. Dagegen stellt es einen prinzipiellen Gewinn dar, die Kovarianzen zur Bewertung der Policy selbst heranzuziehen. Dabei geht man von einer lokalen in eine globale Bewertung über. Sind, als ein Extremfall, die Q-Werte unkorreliert, dann ist die Wahrscheinlichkeit p_{all}, dass jeder wahre Q-Wert tatsächlich unterhalb des Schätzers

$Q_u(s,a)$ liegt, deutlich geringer als p_{single}, die Wahrscheinlichkeit, dass dies für ein einzelnes Zustands-Aktions-Paar zutrifft. Dagegen ist der Wert p_{least}, die Wahrscheinlichkeit, dass dies für mindestens einen Q-Wert zutrifft, deutlich höher als p_{single}. Der Wert p_{single} wird durch ξ und die Gestalt der Verteilung für die Q-Werte determiniert. Dann ergeben sich

$$p_{\text{all}} = p_{\text{single}}^{|S|} \tag{10.23}$$

$$p_{\text{least}} = 1-(1-p_{\text{single}})^{|S|}. \tag{10.24}$$

Im umgekehrten Extremfall, also falls alle Q-Werte maximal korreliert sind, stellt sich die Situation völlig anders dar. Wir erhalten

$$p_{\text{all}} = p_{\text{single}} \tag{10.25}$$

$$p_{\text{least}} = p_{\text{single}}, \tag{10.26}$$

da jedes einzelne Zustands-Aktions-Paar als Repräsentant für alle Zustands-Aktions-Paare stehen kann.

Um eine allgemeine Berechnung der Werte p_{all} und p_{least} zu ermöglichen, also für beliebige Korrelationen, gehen wir im Folgenden von der Näherung aus, dass die Q-Werte normalverteilt sind mit ihren Erwartungswerten Q und der Kovarianz $\mathbf{Cov}(Q,Q)$. Für die wahren Q-Werte $\bar{Q}$ ist also die Dichtefunktion

$$P(\bar{Q}(s_1,a_{1,\max}),\ldots,\bar{Q}(s_{|S|},a_{|A|,\max})) = ce^{-\frac{1}{2}(\bar{Q}-Q)^T\mathbf{Cov}(Q,Q)^{-1}(\bar{Q}-Q)} \tag{10.27}$$

mit geeignetem Normierungsfaktor c. Unter dieser Voraussetzung ergeben sich die gesuchten Werte aus Berechnungen der kumulativen multivariaten Normalverteilung, für die keine analytische Formel, jedoch Heuristiken vorliegen [24]. Wir erhalten demnach

$$\begin{aligned} p_{\text{all}} &= p_{\text{all}}\left(Q,\mathbf{Cov}(Q,Q),Q_u\right) \\ &= c\int_{-\infty}^{Q_u} e^{-\frac{1}{2}\left(\bar{Q}-Q\right)^T\mathbf{Cov}(Q,Q)^{-1}\left(\bar{Q}-Q\right)}d\bar{Q} \end{aligned} \tag{10.28}$$

$$\begin{aligned} p_{\text{least}} &= p_{\text{least}}\left(Q,\mathbf{Cov}(Q,Q),Q_u\right) \\ &= 1-c\int_{Q_u}^{\infty} e^{-\frac{1}{2}\left(\bar{Q}-Q\right)^T\mathbf{Cov}(Q,Q)^{-1}\left(\bar{Q}-Q\right)}d\bar{Q}, \end{aligned} \tag{10.29}$$

wobei hier jeweils Mehrfach-Integrale entsprechend der Anzahl der Zustände zu berücksichtigen sind (die Notation wurde vereinfacht).

Diese Form umfasst schließlich den allgemeinen Fall und damit auch negative Korrelationen, die etwa zu einer Erhöhung von p_{least}, aber zu einer Verringerung von p_{all} führt.

10.2.1.2 Globale Sicherheitsoptimalität

Die oben beschriebene Methode ermöglicht die Bewertung einer globalen Unsicherheit auf der Grundlage lokaler Unsicherheiten und der Nutzung der Kovarianzen, indem man auf p_{all} oder p_{least} zurückgreift. Dies kann sowohl für eine spezifizierte, die optimale als auch die sicherheitsoptimale Policy geschehen. Dagegen ist das Anstreben globaler Sicherheitsoptimalität in Alg. 8 bisher nicht möglich.

10.2.1.2.1 Passive globale Sicherheitsoptimalität Dementsprechend besteht ein erster Schritt zur globalen Sicherheitsoptimalität im Überwachen von p_{all} und p_{least} und Anpassen des Wertes ξ bei entsprechender Veränderung der Wahrscheinlichkeit. Unter der Annahme, dass $P\left(\bar{Q}\left(s_1,a_{1,\max}\right),\ldots,\bar{Q}\left(s_{|S|},a_{|S|,\max}\right)\right)$ eine multivariate Normalverteilung darstellt, wird jedem ξ bei lokaler Betrachtung die Wahrscheinlichkeit

$$p_{\text{single}} = \frac{1+\operatorname{erf}\left(-\xi/\sqrt{2}\right)}{2} \tag{10.30}$$

zugeordnet. Dieser Wert kann nun als Referenzwert zur Fixierung von p_{all} oder p_{least}, je nachdem, was vom Benutzer gefordert wird, herangezogen werden. Der Trade-Off zwischen p_{all} und p_{least} kann durch einen Parameter $0 < \kappa < 1$ gesteuert werden und wir erhalten damit den korrigierten Wert

$$p_{\text{global}} \quad = \quad (1-\kappa)\left(1-(1-p_{\text{single}})^{1/|S|}\right) + \kappa p_{\text{single}}^{1/|S|}, \tag{10.31}$$

der von der initialen Annahme der Unkorreliertheit ausgeht. Für $\kappa = 0$ wird $p_{\text{global}} = p_{\text{least}}$ identifiziert, für $\kappa = 1$ dagegen $p_{\text{global}} = p_{\text{all}}$. Nach jeder Bellman-Iteration wird nun ξ aktualisiert, indem $(1-\kappa)p_{\text{least}} + \kappa p_{\text{all}}$ gegen p_{global} abgeglichen wird. Stellt sich heraus, dass

$$(1-\kappa)p_{\text{least}} + \kappa p_{\text{all}} \quad < \quad p_{\text{global}}, \tag{10.32}$$

dann kann ξ verringert werden, da die geforderte Fehler-Wahrscheinlichkeit noch unterboten wird. Im umgekehrten Fall muss ξ vergrößert werden. Die Änderungsrate kann wieder harmonisch gewählt werden (siehe Abschn. 6.4 und 9.5.1), um gute Konvergenzeigenschaften zu garantieren. In der Konvergenz schließlich wird ein Wert für ξ erreicht, für den p_{global} so groß ist, wie durch die initiale Wahl von ξ implizit gefordert wird.

10.2.1.2.2 Aktive globale Sicherheitsoptimalität Um nun über das Anpassen der Wahrscheinlichkeit p_{global} hinaus aktiv in den Optimierungsvorgang einzugreifen, muss die Aktionsauswahl in jeder Bellman-Iteration modifiziert werden. Als Vorauswahl erfolgt dazu nach wie vor

$$a_{i,\max} = \arg\max_a (Q - \xi\sigma Q)(s_i, a). \tag{10.33}$$

Die dazugehörige Minimalperformance $Q_u(s_i, a_{i,\max})$ ist lokal maximal bzgl. der Aktionen. Die endgültige Auswahl der besten Aktion in jeder Bellman-Iteration erfolgt schließlich durch Fixieren der Grenzen Q_u, Bestimmen von

$$p_{\text{global}} \quad = \quad \left((1-\kappa)\, p_{\text{least}} + \kappa p_{\text{all}}\right)(Q, \mathbf{Cov}(Q,Q), Q_u) \tag{10.34}$$

für alle Aktionen a_j in den Zuständen s_i und Wählen der Aktion, für die p_{global} am kleinsten ist. Mit anderen Worten, es wird die Aktion $a_{i,\text{global,max}}$ als optimal angenommen, für die die Wahrscheinlichkeit für das Unterschreiten der maximalen Minimalperformances (bzgl. der Wahl von κ) minimal ist unter Annahme der sich jeweils aus den verschiedenen Aktionsauswahlen lokal ergebenen Normalverteilungen. Dies ist deshalb zweckmäßig, weil $a_{i,\text{global,max}}$ zu einer höheren Performance als $a_{i,\max}$ (zusammen mit den Performances der anderen Zustände) mit gleicher Wahrscheinlichkeit führen würde. Im Konvergenzfall werden diese Wahrscheinlichkeiten p_{global} für die Aktionen mit $\pi(s_i, a_j) > 0$ gleich groß sein und gerade p_{global} entsprechen, das sich aus dem korrekten ξ ergibt. Die Anpassung von ξ läuft wie oben beschrieben parallel mit.

10.2.1.2.3 Interpretation Bei Optimalsteuerungs- und -regelungsproblemen, für die der Anfangszustand des Systems bekannt oder immer der gleiche ist, kann man sich auf lokale Sicherheitsoptimalität zurückziehen. Ist jedoch der Anfangszustand unbekannt oder hängt von unbekannten Faktoren ab, dann ist die globale Sicherheitsoptimalität von entsprechend großer Bedeutung.

Für ein sicherheitskritisches System, etwa ein Kernkraftwerk, das in verschiedenen Zuständen anlaufen kann, sollte $\kappa = 0$ gewählt werden. Damit kann mit p_{global} garantiert werden, dass die berechnete Mindestperformance auch tatsächlich erzielt wird. Auch in Wettkampfsituationen ist diese Einstellung sinnvoll, wenn wenige Informationen über die anderen Teilnehmer und deren Fitness bekannt sind. Die Wahl von $\kappa = 0$ führt tendenziell zu einer erhöhten Korreliertheit der einzelnen Q-Werte, da durch die stärkere Bindung der Q-Werte untereinander negative Einzelfälle unwahrscheinlicher werden.

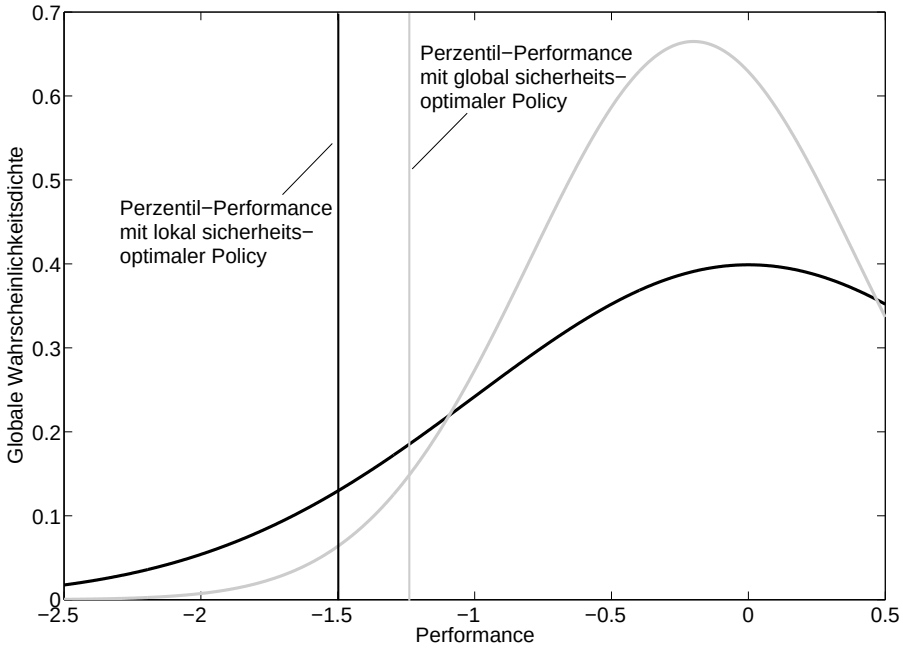


Abbildung 10.1: Illustration der Funktionsweise des Algorithmus zum Anstreben globaler Sicherheitsoptimalität. Im hier dargestellten Beispiel entspricht die schwarz gezeichnete Verteilung der globalen Performance-Verteilung für die lokale sicherheitsoptimale Aktion. Die durch die graue Kurve repräsentierte Aktion kann jedoch mit einer höheren Wahrscheinlichkeit die gleiche lokale Performance erzielen. Daher muss auch ihre lokale Performance bei gleicher Wahrscheinlichkeit größer sein. Bei der Auswahl der optimalen Aktion während der Iteration wird also die grau repräsentierte Aktion gewählt.

Die Einstellung $\kappa = 1$ ist eher in Systemen zweckmäßig, die mehrmals aus verschiedenen Startzuständen anlaufen und wenigstens einmal die Mindestperformance überschreiten sollen. Dies ist bei weniger sicherheitskritischen Systemen oder wiederholbaren Versuchen sinnvoll, darüber hinaus bei Wettkämpfen, die sich häufig wiederholen. Dem Wettkampfteilnehmer wird dann garantiert, dass wenigstens einmal ein gutes Ergebnis erzielt werden kann. Die Wahl $\kappa = 1$ führt zu weniger korrelierten Q-Werten, da das Risiko, dass $\bar{Q}(s, a)$ unterhalb von $Q_u(s, a)$ liegt, auf mehrere Zustände verteilt wird.

10.2.2 Berücksichtigung höherer Momente

Bei den bisherigen Betrachtungen gingen wir davon aus, dass wir die Verteilungen der Transitionswahrscheinlichkeiten, Rewards und schließlich auch der Q-Werte ausschließlich durch Erwartungswert und Varianz sowie der Kovarianz der verschiedenen Größen beschreiben. Dies kann in der Praxis eine Einschränkung sein, da die Symmetrieeigenschaften und die Wölbung sowie weitere statistischen Kenngrößen der Verteilungen für eine differenzierte Betrachtung nicht berücksichtigt werden können.

Eine Implementierung einer dazu in diesem Abschnitt vorgestellten verallgemeinerten Unsicherheitspropagation ist jedoch noch zu entwickeln und zu erproben. Wir erläutern hier die grundsätzliche Idee und ihre Machbarkeit.

10.2.2.1 Schiefe und Kurtosis

Schiefe und Kurtosis sind statistische Größen, die sich durch sogenannte zentrale Momente der Verteilung ausdrücken lassen. Man definiert das zentrale Moment $k$ter Ordnung als

$$\mu_k(X) \quad = \quad \mathbf{E}\big((X - \mathbf{E}X)^k\big). \tag{10.35}$$

Das zentrale Moment erster Ordnung ist also stets 0, während μ_2 gerade die Varianz von X beschreibt. Schiefe und Kurtosis ergeben sich aus drittem und viertem zentralen Moment als

$$\mathbf{Sk}(X) \quad = \quad \frac{\mu_3}{\sigma^3} \tag{10.36}$$

$$\mathbf{Kur}(X) \quad = \quad \frac{\mu_4}{\sigma^4}. \tag{10.37}$$

Manchmal wird statt der Kurtosis auch der Exzess $\mathbf{Kur}(X) - 3$ angegeben. Die Schiefe beschreibt die Asymmetrie einer Verteilung. Für $\mathbf{Sk}(X) > 0$ ist die Verteilung rechtsschief, für $\mathbf{Sk}(X) < 0$ entsprechend linksschief. Für unsere Zwecke erlaubt die Berücksichtigung der Schiefe eine wesentlich bessere Abschätzung der Unsicherheit, da wir die Performance stets nach unten abschätzen wollen. Linksschief verteilte Q-Werte können also für $\xi > 0$, trotz hoher Varianz zu einer günstigeren Performance-Abschätzung führen, da sich der Modalwert der Verteilung auf der rechten Seite des Erwartungswertes befindet und die Höhe der Varianz vor allem aus der Volatilität nach rechts ergibt. Umgekehrt erfordert eine rechtsschiefe Verteilung eine konservativere Abschätzung.

Die Kurtosis dagegen beschreibt die Gaußartigkeit oder die Wölbung einer Verteilung. Für $\mathbf{Kur}(X) > 3$ ist die Verteilung steilgipflig, also spitzer als die Normalverteilung, während sie für $\mathbf{Kur}(X) < 3$ flachgipflig und damit abgeflachter als die Normalverteilung ist. Dies hat unmittelbare Auswirkungen auf den Zusammenhang zwischen ξ und p_{single}. Je steilgipfliger die Verteilung ist, desto konservativer muss die Performance-Abschätzung bei betragsmäßig höheren Werten für ξ sein, da sie in diesem Fall über ausgeprägte Enden verfügt.

Natürlich sind auch zentrale Momente höherer Ordnung von Bedeutung. Wegen der schwierigeren Interpretierbarkeit und der algorithmischen Komplexität der Propagation höherer zentraler Momente konzentrieren wir uns ergänzend zur Varianz nur auf Schiefe und Kurtosis. Neben der Kovarianz lassen sich die höheren Verbundmomente $k$ter Ordnung durch

$$\mu_k\left(X_1, \ldots, X_n\right) \quad = \quad \mathbf{E}\Big(\left(X_1 - \mathbf{E}X_1\right)^{k_1} \cdots \left(X_n - \mathbf{E}X_n\right)^{k_n}\Big) \tag{10.38}$$

mit $k = \sum_{i=1}^{n} k_i$ bestimmen.

Bei der Unsicherheitspropagation muss also neben der Propagation der Erwartungswerte und der Kovarianzen zusätzlich auch Koschiefe und Kokurtosis propagiert werden. Dass dies, unter den gleichen Voraussetzungen wie für die Kovarianz, tatsächlich möglich ist, folgt aus der Linearität der ersten zentralen Momente bzgl. ihrer Zufallsvariablen. Für statistisch unabhängige Zufallsvariablen X und Y ist

$$a\mu_k(X) + b\mu_k(Y) \quad = \quad \mu_k(aX + bY) \tag{10.39}$$

auf Grund ihres Zusammenhangs zu den sogenannten Kumulanten, für $k \leq 3$. Ist $k = 2$, dann muss lediglich Unkorreliertheit vorausgesetzt werden, für $k = 1$ gilt stets Linearität. Für $k \geq 4$, also für die Kurtosis und alle höheren Momente gilt dieser Zusammenhang nur näherungsweise.

Das Prinzip der Propagation höherer Verbundmomente $k$ter Ordnung für die Funktion $f : \mathbb{R}^m \to \mathbb{R}^n$ geschieht dann mittels

$$\begin{aligned} \mathbf{Comoment}(f_{i_1}, \ldots, f_{i_k}) \quad = \quad & \sum_{j_1=1}^{n} \cdots \sum_{j_k=1}^{n} \frac{\partial f_{i_1}}{\partial \mathbf{x}_{j_1}} \cdots \frac{\partial f_{i_k}}{\partial \mathbf{x}_{j_k}} \\ & \mathbf{Comoment}(\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_k}). \end{aligned} \tag{10.40}$$

Entscheidend ist nun wieder die Bestimmung von Q_u, entweder nach der Identifikation der Policy oder, um Sicherheitsoptimalität zu erzielen, nach jeder Bellman-Iteration zur Bestimmung der lokal optimalen Aktion. Dazu ziehen wir uns zunächst wieder auf die Diagonalelemente der Kovarianzmatrix zurück. Es ergibt sich

$$\begin{aligned} Q_u(s,a) &= Q(s,a) - g\,(\mathbf{Cov}(Q(s,a),Q(s,a)), \\ &\quad \mathbf{Cosk}(Q(s,a),Q(s,a),Q(s,a)), \\ &\quad \mathbf{Cokur}(Q(s,a),Q(s,a),Q(s,a),Q(s,a)),\xi)\,, \end{aligned} \tag{10.41}$$

wobei g vom geschätzten Erwartungswert $Q(s,a)$, seiner Varianz, Schiefe und Kurtosis sowie dem Wert ξ abhängt. Die Funktion g kann etwa mit Hilfe der momenterzeugenden Funktion berechnet werden.

Kapitel 11

Benchmarks und industrielle Anwendungen

Zur Demonstration der Realisierbarkeit der Sicherheitsoptimalität und ihrer Bedeutung für industrielle Problemstellungen durch die hier vorgestellten Methoden werden auch in diesem Kapitel Ergebnisse auf Benchmarks und Simulationen von Gasturbinen beschrieben und erläutert. Wir konzentrieren uns dabei auf die Anwendung der Unsicherheitspropagation zusammen mit der diskreten Bellman-Iteration. Ferner zeigen wir einige Ergebnisse im Zusammenhang mit der Ausnutzung der vollständigen Kovarianzmatrix.

11.1 Demonstration der Qualitätssicherung

Zur generellen Demonstration der Eigenschaften sicherheitsoptimaler Policies haben wir die verschränkte Iteration auf einem festen Datensatz für zwei einfache Klassen von MDP angewendet. Damit noch eine interpretierbare grafische Darstellung und eine algorithmische Realisierung möglich sind, betrachten wir im ersten Fall ein einfaches Spielautomaten- bzw. Banditenproblem (ein Zustand, zwei Aktionen) und im zweiten Fall MDP mit zwei Zuständen und zwei Aktionen.

Wir gehen dabei von einer festen beobachteten Trajektorie aus. Beim zweiten Benchmark-Problem wurden etwa folgende Transitionen als beobachtet angenommen.

$$\mathbf{s} = (1,1,1,1,1,2,2,2,2,2,2) \tag{11.1}$$

$$\mathbf{a} = (1,1,2,2,1,1,1,2,2,2) \tag{11.2}$$

$$\mathbf{r} = (1.35,1,1,1,1,1,0,0,1,-1) \tag{11.3}$$

Auf der Grundlage dieser Beobachtungen wurde die verschränkte Bellman-Iteration für verschiedene Werte für ξ durchgeführt, die jeweils zu einem bestimmten klar definierten Ergebnis führen, also einer Policy π^{ξ}, die, abgesehen von den Beobachtungen, nur von ξ abhängt. Für die Berechnung von Q^{ξ} und π^{ξ} wurde der Bayes'sche Ansatz gewählt und ein Prior festgelegt, für den die Transitionswahrscheinlichkeiten ausgehend von einem bestimmten Zustand unter Anwendung einer bestimmten Aktion jeweils als multinomial verteilt angenommen werden. Ihre Parameter unterliegen einer uniformen a priori Verteilung, die sich auch als Beta-Verteilung mit $\alpha = \beta = 1$ ausdrücken lässt. Die Beta-Verteilung ist identisch mit der Dirichlet-Verteilung im zweidimensionalen Fall, dabei werden α mit α_1 und β mit α_2 identifiziert.

Die Transitionswahrscheinlichkeiten für verschiedene Zustands-Aktions-Paare werden als statistisch unabhängig betrachtet. Die Rewards nehmen wir als normalverteilt mit fester Varianz $\sigma_0 = 1$ an mit ebenfalls normalverteiltem Prior mit Erwartungswert $\mu = 0$ und Varianz $\sigma = 1$. Beide a priori Verteilungen sind konjugierte Prior für die entsprechenden Parameter

und lassen sich daher ohne großen Aufwand zusammen mit konkreten Beobachtungen in a posteriori Verteilungen überführen.

Die Schätzer für P und R sowie die initiale Kovarianzmatrix C^0 wurden genau so besetzt, dass sie den auf die oben beschriebene Weise gewonnenen a posteriori Verteilungen entsprechen. Parallel dazu wurde über die Menge aller möglichen MDP, genauer über die Menge aller den MDP beschreibenden Parameter gemäß der oben beschriebenen a priori Verteilungen gesampelt. Jeder so gewonnene MDP wurde mit bestimmten Policies π getestet und die durchschnittliche Performance

$$p(\pi) \quad = \quad \frac{1}{P(\mathbf{s}, \mathrm{a}, \mathrm{r})} \int_M P(\mathbf{s}, \mathrm{a}, \mathrm{r}|M) P(M) V_M^\pi(s_0) dM \tag{11.4}$$

geschätzt. Dazu wird eine Reihe möglicher MDP M mit der a priori Wahrscheinlichkeit $P(M)$ gesampelt und zur Bestimmung der mittleren Performance mit dem Likelihood $P(\mathbf{s}, \mathrm{a}, \mathrm{r}|M)$ der beobachteten Daten gewichtet. Auf diese Weise gewinnt man die a posteriori Verteilung, die die beobachtete Trajektorie berücksichtigt.

11.1.1 Ergebnisse

In Abb. 11.1 sind diese a posteriori Verteilungen der Performances verschiedener Policies dargestellt. Offensichtlich nehmen Erwartungswert und Varianz für jede Policy unterschiedliche Werte an. Die erwartungswert-optimale Policy hat den höchsten Erwartungswert, während sichere, stochastische Policies eine geringere Varianz aufweisen und die Wettkampf-Policy über eine breitere Verteilung verfügt. Jede dieser Eigenschaften ist gerade Voraussetzung für die für den jeweiligen Policy-Typ erwünschte Aufgabe (siehe Abschn. 9.6).

Die Abbn. 11.2 und 11.3 zeigen die minimalen Performances für die beiden Benchmark-Problemklassen unter Anwendung der in Abb. 11.1 dargestellten (für Abb. 11.3, gleiche Farben, gleiche Linienstile) und weiterer Policies in Abhängigkeit von der Wahrscheinlichkeit des Auftretens der MDP. Die kumulative Verteilung der erreichbaren Performances (Abb. 11.1 zeigt die entsprechende Dichtefunktion) ist gerade die Umkehrfunktion der dargestellten Graphen. Sie ermöglicht einen Vergleich der bzgl. des Samplings über die MDP erwarteten Policy-Performances an verschiedenen Perzentilen. So ist etwa in Abb. 11.3 deutlich zu erkennen, dass am 10-Prozent-Perzentil die vollständig stochastische Policy performanter ist als jede andere dargestellte, während am 90-Prozent-Perzentil eine andere deterministische Policy die höchste Performance erzielt als am 50-Prozent-Perzentil.

In Tab. 11.1 werden exemplarisch für die Abb. 11.3 die ermittelten Policies für verschiedene ξ aufgelistet. Sie stimmen mit den jeweils durch das Sampling als maximal bestimmten Policies näherungsweise überein. Für steigendes ξ, also sinkendem Perzentil, werden etwa zunächst die Aktionen im ersten Zustand stochastisch und später auch die Aktionen im zweiten Zustand. Für fallendes ξ wechselt ab einem bestimmten Punkt die deterministische Policy in der Aktion im ersten Zustand, während die Aktion im zweiten Zustand bestehen bleibt. Jede dieser Beobachtungen lässt sich sowohl in der Grafik als auch in der Tabelle nachvollziehen.

11.2 Steigerung der Informationseffizienz und Qualitätssicherung an Benchmarks

Wie das in Abschn. 9.6 beschriebene Randphänomen überwunden werden kann, zeigen wir exemplarisch an einem konstruierten Benchmark-Problem. Die UPRL mit der verschränkten Iteration trägt dort zu einer Steigerung der Informationseffizienz bei, so dass bei gleicher Anzahl an Beobachtungen die Performance der ermittelten Policies erhöht wird.

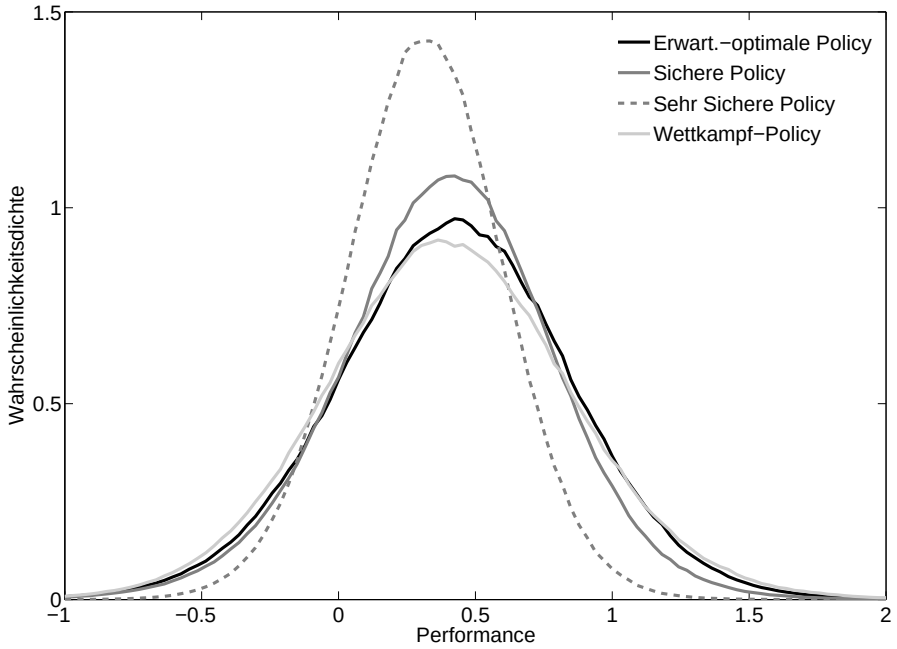


Abbildung 11.1: Performance-Verteilung verschiedener Policies für einfache MDP mit zwei Zuständen und zwei Aktionen. Die verschiedenen Graphen zeigen die Verteilungen, die näherungsweise Normalverteilungen sind, über die Performances auf möglichen MDP, die von unterschiedlichen (stochastischen) Policies erzielt werden. Der Erwartungswert der erwartungswert-optimalen Policy ist am höchsten, während die sehr sichere Policy, die die stärkste Stochastizität aufweist, über die kleinste Varianz verfügt und daher unterhalb eines gewissen Perzentils die beste Perzentil-Performance erzielt.

Tabelle 11.1: Ermittelte sicherheitsoptimale Policies für unterschiedliche Konfidenzlevel, obigen Datensatz (Gln. 11.1, 11.2 und 11.3) und dem dort angenommenen Prior zur Beschreibung von MDP mit zwei Zuständen und zwei Aktionen. Zusätzlich werden die durch die Q-Funktion ausgedrückte geschätzte Performance und die Entropie der bestimmten Policies dargestellt. Die Einträge sind konsistent mit der Abb. 11.3, das heißt die ermittelten Policies stimmen näherungsweise mit denen überein, die tatsächlich an den jeweiligen Perzentilen sicherheitsoptimal sind.

ξ	**øGesch. Performance**	$\pi(\mathbf{1},\mathbf{1})$	$\pi(\mathbf{1},\mathbf{2})$	$\pi(\mathbf{2},\mathbf{1})$	$\pi(\mathbf{2},\mathbf{2})$	**Entropie**
4	−0.663	0.57	0.43	0.52	0.48	0.992
3	−0.409	0.58	0.42	0.55	0.45	0.987
2	−0.161	0.59	0.41	0.60	0.40	0.974
1	0.106	0.61	0.39	0.78	0.22	0.863
2/3	0.202	0.67	0.33	1	0	0.458
0	0.421	1	0	1	0	0
−2/3	0.651	1	0	1	0	0
−1	0.762	1	0	1	0	0
−2	1.103	1	0	1	0	0
−3	1.429	0	1	1	0	0
−4	1.778	0	1	1	0	0

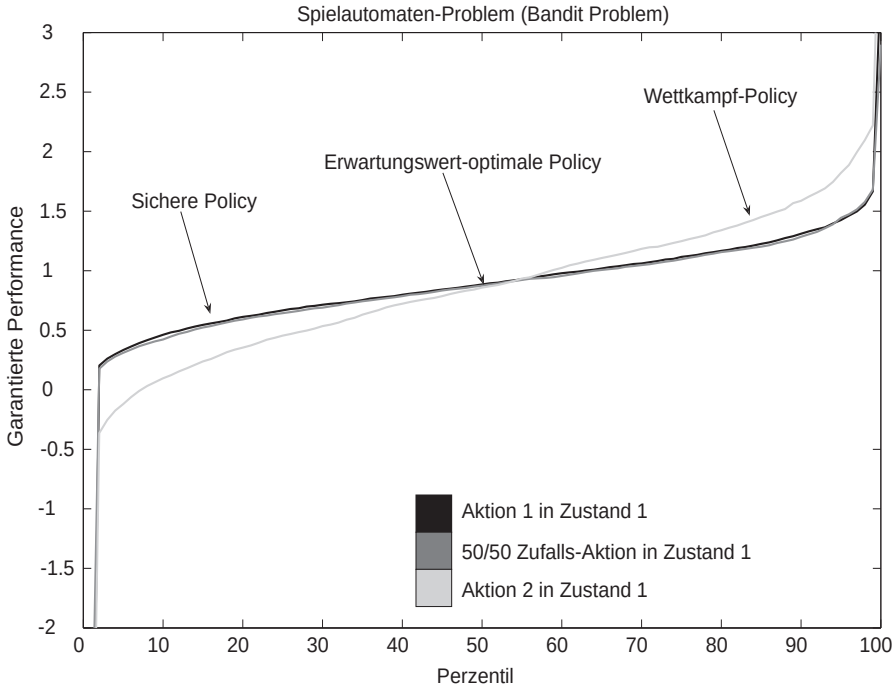


Abbildung 11.2: Perzentil-Performance für einfache Spielautomatenprobleme. Die verschiedenen Graphen zeigen die minimale Performance, die von unterschiedlichen (stochastischen) Policies erzielt wird. Dabei ist der Wert am Perzentil x die von wenigstens x Prozent der gesampelten MDP erzielte Performance. Die Graustufen bestimmen, welche Aktion in dem einzigen Zustand gewählt werden soll.

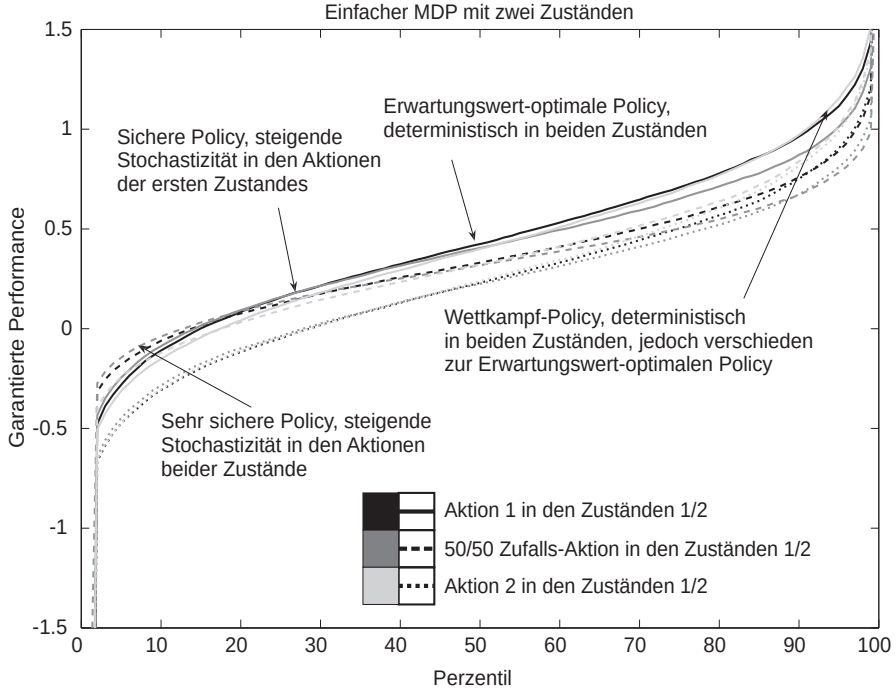


Abbildung 11.3: Perzentil-Performance für einfache MDP mit zwei Zuständen und zwei Aktionen. Die verschiedenen Graphen zeigen die minimale Performance, die von unterschiedlichen (stochastischen) Policies erzielt wird. Dabei ist der Wert am Perzentil x die von wenigstens x Prozent der gesampelten MDP erzielte Performance. Die Graustufen und der Linienstil bestimmen, welche Aktionen in den beiden Zuständen gewählt werden sollen.

11.2.1 Das Bogenschießen-Problem

Wir betrachten dazu ein Bogenschießen-Problem, welches per Konstruktion das oben beschriebene Randphänomen aufweist. Der Zustandsraum repräsentiert den Bereich einer von einem Schützen anvisierbaren Zielscheibe (siehe Abb. 11.4). Der Aktionsraum besteht aus fünf Aktionen und ermöglicht das Bewegen der Pfeilspitze in alle vier Himmelsrichtungen (bezogen auf die Zielscheibe) sowie das Abschießen des Pfeiles. Wir simulierten dieses Benchmark-Problem mit zwei verschiedenen zu Grunde liegenden MDP. Im deterministischen Fall ist die Bewegung der Pfeilspitze deterministisch, im stochastischen Fall dagegen werden die Auswirkungen einer Aktion mit 25-prozentiger Wahrscheinlichkeit auf die einer anderen Aktion übertragen. In beiden MDP ist jedoch die Wahrscheinlichkeit eines Treffers im Falle eines Schusses gleich und wird durch die in Abb. 11.4 angegebenen Werte definiert. Die höchste Trefferwahrscheinlichkeit liegt in der Mitte der Zielscheibe vor.

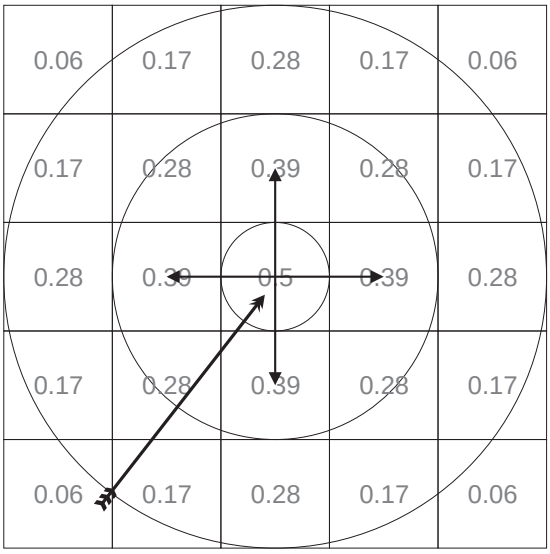


Abbildung 11.4: Illustration des Bogenschießen-Benchmarks. Die Grafik zeigt die Zielscheibe, die aus 25 Zuständen besteht, zusammen mit den zugehörigen Trefferwahrscheinlichkeiten. Der Aktionsraum besteht aus den Aktionen Links, Rechts, Oben und Unten sowie Abschießen des Pfeiles. Der Agent vereinnahmt nur bei einem Treffer einen Reward von 1.

Der Vorgang der Exploration erfolgt zufällig jeweils von der Mitte der Zielscheibe ausgehend über 25 Schritte. Dies führt dazu, dass der Rand der Zielscheibe nur selten exploriert wird, jedoch im Einzelfall an einem der Randzustände ein Treffer verzeichnet werden kann. Dies wiederum führt bei der Anwendung klassischen Reinforcement-Learnings zu der fälschlichen Vermutung des Agenten, dort die höchste Trefferwahrscheinlichkeit, damit hohe Rewards zu erzielen und schließlich eine Policy vorzuschlagen, die von dort aus den Pfeil schießt.

Die Kovarianzmatrix wurde in beiden Fällen nach dem frequentistischen Paradigma für die Transitionswahrscheinlichkeiten bestimmt. Für den deterministischen MDP entspricht dies dem optimalen Prior, da mit einer Beobachtung der entsprechende Schätzer bereits korrekt ist. Die Rewards werden ebenfalls frequentistisch behandelt.

11.2.2 Ergebnisse

In der Tab. 11.2 sind die erzielten mittleren Rewards für das Bogenschießen-Benchmark-Problem mit den beiden verschiedenen Parametersätzen aufgelistet. Dem liegen jeweils 50 Versuche zu Grunde, was zu zweistelliger Präzision führt. Die Ergebnisse zeigen, dass sich die Performance mit zunehmendem ξ tatsächlich bis zu einem Maximalwert erhöht und dann

schnell abfällt. Die Position des Maximums hängt von der Anzahl der Beobachtungen ab. Je mehr Transitionen beobachtet werden, desto höher liegt das Maximum. Dies kann durch die parallel reduzierte Unsicherheit erklärt werden. Mit einer wachsenden Zahl an Beobachtungen entspricht eine gleich bleibende Performance-Korrektur einem höheren Konfidenzlevel.

Die theoretische Maximalperformance für den stochastischen MDP liegt bei 0.31, für den deterministischen MDP beträgt sie 0.5. Diese Werte können mit 2500 Beobachtungen für $1 \leq \xi \leq 2$ im stochastischen Fall und für $3 \leq \xi \leq 4$ im deterministischen Fall im Durchschnitt erreicht werden. Interessant ist die gute Performance für nur 100 Beobachtungen im deterministischen Fall bei kleinen Werten für ξ. Wir vermuten, dass dieses Artefakt dadurch zu Stande kommt, dass die wenigen Beobachtungen sich fast ausschließlich in der Nähe der Mitte der Zielscheibe aufhalten und daher andere Zustände auch von der durch die Bellman-Iteration ermittelten Policy nicht besucht werden.

Tabelle 11.2: Durchschnittlicher Reward für das Bogenschießen-Benchmark-Problem. Für den stochastischen MDP wird eine frequentistische, für den deterministischen eine deterministische Modellierung der Transitionswahrscheinlichkeiten herangezogen. Die Werte beziehen sich auf unterschiedlich große Beobachtungsdatensätze und verschiedene Konfidenzlevel. Eine Steigerung der Performance durch eine Wahl von $\xi > 0$ ist deutlich zu erkennen.

MDP	**# Beob.**	$\xi = -\frac{1}{2}$	$\xi = \mathbf{0}$	$\xi = \frac{1}{2}$	$\xi = \mathbf{1}$	$\xi = \mathbf{2}$	$\xi = \mathbf{3}$	$\xi = \mathbf{4}$	$\xi = \mathbf{5}$
sto-	100	0.15	0.14	**0.16**	0.13	0.05	0.05	0.04	0.04
chas-	500	0.15	0.17	0.20	**0.25**	0.22	0.10	0.05	0.04
tisch	1000	0.18	0.21	0.26	**0.29**	0.27	0.22	0.11	0.07
	2500	0.26	0.27	0.29	**0.31**	**0.31**	0.30	0.28	0.24
deter-	100	0.32	0.35	**0.38**	0.23	0.17	0.12	0.11	0.09
minis-	500	0.28	0.32	0.38	0.39	**0.41**	0.27	0.18	0.11
tisch	1000	0.32	0.35	0.41	0.44	**0.45**	0.44	0.30	0.14
	2500	0.42	0.44	0.46	0.48	0.49	**0.50**	**0.50**	0.48

11.2.3 Berücksichtigung der vollständigen Kovarianzmatrix

Für das gleiche Benchmark-Problem haben wir auch die erweiterte sicherheitsoptimale Bellman-Iteration getestet, die die vollständige Kovarianzmatrix zur Bestimmung der optimalen Aktionen benutzt. Wir betrachten den Fall $\kappa = 0$, der für die Praxis von größerer Bedeutung ist und simulierten dazu den Explorationsvorgang für das stochastische Bogenschießen-Benchmark-Problem unter Anwendung verschiedener Werte für ξ.

Tab. 11.3 zeigt die durchschnittlich geschätzte Performance, also den durchschnittlichen Q-Wert und die Wahrscheinlichkeit p_{least}, die von einer multivariaten Normalverteilung ausgeht und von ξ und der Gestalt der Kovarianzmatrix abhängt. Diese globale Fehlerwahrscheinlichkeit fällt tendenziell für die kovarianzberücksichtigende Iteration, da sie anstelle von p_{single} optimiert wird. Durch die iterative Veränderung der Kovarianzen wird dadurch auch das lokale ξ erhöht (für $\kappa = 1$ würde ξ kleiner werden), was bei gleicher garantierter Performance einen Vorteil gegenüber dem Standard-Verfahren darstellt. Zum besseren Vergleich haben wir daher zusätzlich die den Fehlerwahrscheinlichkeiten p_{least} zugeordneten realen Werte für ξ aufgeführt, damit die Konfidenzlevel verglichen werden können.

Die Werte für p_{least} bei der Standard-Unsicherheitspropagation liegen in der Tendenz höher als die der kovarianzberücksichtigenden Bellman-Iteration, sie sind für $\xi = 1$ sogar fast doppelt so groß. Bei sehr kleinen Wahrscheinlichkeiten kann der Wert auch kleiner sein. Die durchschnittliche Performance der Standard-Iteration unterscheidet sich dagegen nur gering von der kovarianzberücksichtigenden und liegt fast immer leicht darüber. Dies muss auch so sein, da die Standard-Iteration schließlich für das angegebene ξ die höchste garantierte Performance anstrebt. Die Mittelwerte in der rechten Spalte verdeutlichen den Gewinn, der durch das kovarianzberücksichtigende Verfahren erzielt wird. Es zeigt sich, dass gegen einen geringen Performance-Verlust eine günstigere globale Fehlerwahrscheinlichkeit erreicht werden kann.

Tabelle 11.3: Wahrscheinlichkeit für das Nicht-Erreichen der Minimalperformance (Fehlerwahrscheinlichkeit) an mindestens einem Zustand und die durchschnittliche erwartete Performance für die Unsicherheitspropagation mit und ohne Berücksichtigung der vollständigen Kovarianzmatrix. Das ebenfalls dargestellte reale ξ entspricht dem Konfidenzlevel, das sich umgekehrt aus der Fehlerwahrscheinlichkeit ergeben würde. In der am weitesten rechts stehenden Spalte sind jeweils die Durchschnittswerte für die Performance und das reale ξ aufgeführt.

	ξ								
Methode	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	Ø
	Geschätzte Performance								
Standard	4.3	4.1	3.6	3.2	2.9	2.5	2.3	1.8	3.1
Kovarianz	3.9	3.8	3.5	3.3	2.9	2.5	2.2	1.6	3.0
	Fehlerwahrscheinlichkeit p_{least}								
Standard	2.50 10^{-1}	1.05 10^{-1}	3.24 10^{-2}	7.44 10^{-3}	1.57 10^{-3}	3.12 10^{-4}	3.20 10^{-5}	4.32 10^{-5}	
Kovarianz	1.34 10^{-1}	5.85 10^{-2}	2.50 10^{-2}	7.17 10^{-3}	1.58 10^{-3}	2.72 10^{-4}	3.69 10^{-5}	4.05 10^{-6}	
	Reales ξ **bzgl.** p_{least}								
Standard	0.68	1.25	1.85	2.44	2.95	3.42	4.00	4.45	2.63
Kovarianz	1.11	1.57	1.96	2.45	2.95	3.46	3.96	4.46	2.74

11.3 Steigerung der Informationseffizienz bei der Steuerung von Gasturbinen

Bei den in Abschn. 8.2 beschriebenen Gasturbinen-Simulationen können wir davon ausgehen, dass das Randphänomen ebenfalls eine Rolle spielt. Die Gasturbinen werden überwiegend in der Nähe von Arbeitspunkten betrieben und exploriert, an denen bereits eine hohe Performance erzielt werden kann. Sie sind daher dem Bogenschießen-Benchmark qualitativ ähnlich.

11.3.1 Beschreibung der Parameter

Für eine Diskretisierung des Zustandsraumes wurde dieser zunächst auf vier Dimensionen reduziert. Dies geschah unter Anwendung rekurrenter neuronaler Netze zur Dimensionsreduktion wie in Abschn. 8.2.4 beschrieben. Die Diskretisierung erfolgte dann jeweils mit drei verschiedenen Präzisionsgraden. Da tatsächlich nur ein Teil der möglichen diskreten Zustände vom Agenten während der Exploration besucht wurde, konnte die Anzahl der Zustände weiter reduziert und dadurch eine statistisch signifikante Auswertung durchgeführt werden. Auf Grund des unterschiedlichen Besetzungsgrades der Zustände der beiden Turbinen-Simulationen und der Skalierung der Unsicherheitspropagation, mussten wir auf unterschiedliche Diskretisierungsstufen zurückgreifen.

- Brennkammer-Simulation
  - grob, $4^4 = 256$ Zustände, 33 echte Zustände (13% Besetzung)
  - medium, $5^4 = 625$ Zustände, 55 echte Zustände, (9% Besetzung)
  - fein, $6^4 = 1296$ Zustände, 83 echte Zustände, (6% Besetzung)
- Emissionsreduktions-Simulation
  - grob, $2^4 = 16$ Zustände, 14 echte Zustände, (88% Besetzung)
  - medium, $3^4 = 81$ Zustände, 51 echte Zustände, (63% Besetzung)
  - fein, $4^4 = 256$ Zustände, 101 echte Zustände, (39% Besetzung)

11.3.2 Ergebnisse

In den Tabn. 11.4 und 11.5 sind die erreichten mittleren Rewards für die beiden Gasturbinensimulationen aufgelistet, die von Policies erzielt wurden, die auf unterschiedlichen Datensätzen trainiert wurden. Die Ergebnisse basieren ebenfalls auf jeweils 50 Experimenten, was zu einer drei- bzw. zweistelligen Präzision führt. Neben dem frequentistischen Paradigma

Tabelle 11.4: Durchschnittlicher Reward und Entropie der Policies für die Brennkammer-Simulation mit frequentistischer und Bayes'scher Modellierung der Transitionswahrscheinlichkeiten mit uniformem Prior unter Berücksichtigung von jeweils 10000 Beobachtungen für unterschiedliche Diskretisierungsstufen und Konfidenzlevel. Die Entropie ist ein Maß für die Stochastizität der Policy, die mit steigendem ξ ebenfalls ansteigt. Da die maximale Entropie bei 2.3219 liegt, sind die Policies aber auch bei hohen Werten für ξ noch stark determiniert.

Diskr.	$\xi = \mathbf{0}$	$\xi = \frac{\mathbf{1}}{\mathbf{2}}$	$\xi = \mathbf{1}$	$\xi = \mathbf{2}$	$\xi = \mathbf{3}$	$\xi = \mathbf{4}$	$\xi = \mathbf{5}$	$\xi = \mathbf{6}$	$\xi = \mathbf{7}$
Performance/frequentistische Modellierung									
grob	0.736	0.758	0.770	0.815	0.837	0.848	0.855	0.857	**0.858**
medium	0.751	0.769	0.784	0.816	**0.833**	0.830	0.815	0.815	0.803
fein	0.767	0.785	0.800	0.826	0.837	**0.840**	0.839	0.836	0.831
Entropie/frequentistisch									
grob	0	0.053	0.093	0.164	0.223	0.259	0.289	0.324	0.352
medium	0	0.067	0.128	0.224	0.297	0.335	0.354	0.375	0.389
fein	0	0.063	0.099	0.168	0.210	0.236	0.253	0.270	0.283
Performance/Bayes'sche Modellierung									
grob	0.720	0.767	0.814	0.848	0.851	**0.854**	**0.854**	0.848	0.844
medium	0.713	0.731	0.749	0.777	**0.787**	0.780	0.771	0.739	0.737
fein	0.735	0.773	0.789	**0.800**	**0.800**	0.786	0.779	0.750	0.740

Tabelle 11.5: Durchschnittlicher Reward für die Emissionsreduktions-Simulation mit frequentistischer und Bayes'scher Modellierung der Transitionswahrscheinlichkeiten mit uniformem Prior unter Berücksichtigung von jeweils 10000 Beobachtungen für unterschiedliche Diskretisierungsstufen und Konfidenzlevel.

Diskr.	$\xi = \mathbf{0}$	$\xi = \frac{\mathbf{1}}{\mathbf{2}}$	$\xi = \mathbf{1}$	$\xi = \mathbf{2}$	$\xi = \mathbf{3}$	$\xi = \mathbf{4}$	$\xi = \mathbf{5}$	$\xi = \mathbf{6}$
Frequentistische Modellierung								
grob	0.056	0.081	0.093	0.122	0.153	0.170	0.177	**0.184**
medium	0.112	0.104	0.151	**0.250**	0.238	0.249	0.246	0.249
fein	−0.03	0.07	**0.18**	0.13				
Bayes'sche Modellierung								
grob	0.033	0.088	0.115	0.132	0.133	0.140	0.154	**0.158**
medium	0.161	0.195	0.212	**0.230**	0.226	0.221	**0.230**	0.203

zur Bestimmung der Kovarianzmatrizen ist hier alternativ ein uninformierter Dirichlet-Prior für die Transitionswahrscheinlichkeiten zur Anwendung gekommen. Wir wählten die uniforme Verteilung, für die die Parameter der Dirichlet-Verteilung gemäß $\forall i : \alpha_i = 1$ gewählt werden müssen und die eine maximale Entropie aufweist. Die Rewards werden ebenfalls mit uninformiertem Prior geschätzt. Dabei wird eine Normal-Gamma-Verteilung mit $\sigma = \infty$ und $\alpha = \beta = 0$ als a priori Verteilung angenommen. Für das frequentistische Modell bei der Brennkammer-Simulation haben wir in Tab. 11.4 exemplarisch zusätzlich die Entropie der Policies dargestellt, die sich gemäß

$$\text{Entropie}(\pi) \quad = \quad -\frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{j=1}^{|A|} \pi(s_i, a_j) \log(\pi(s_i, a_j)) \tag{11.5}$$

berechnen lässt und ein Maß für die Stochastizität der ermittelten Policies ist. Man beachte, dass die maximal mögliche Entropie bei der uniformen Policy angenommen würde und in

diesem Fall also $-\log\left(\frac{1}{5}\right) = 2.3219$ beträgt. Die ermittelten Werte zeigen, dass die Stochastizität der Policies mit steigendem ξ tatsächlich zunimmt, der hohe Abstand zur uniformen Policy jedoch noch einen hohen Bestimmtheitsgrad der Policies zeigt. In der Tat verteilt sich die Stochastizität meist nur über einen kleinen Teil der Aktionen.

Tabelle 11.6: Reward für die Brennkammer-Simulation mit frequentistischer und Bayes'scher Modellierung der Transitionswahrscheinlichkeiten mit uniformem Prior unter Berücksichtigung von 100000 Beobachtungen für unterschiedliche Diskretisierungsstufen und Konfidenzlevel für ein exemplarisches Experiment.

Diskr.	$\xi = \mathbf{0}$	$\xi = \mathbf{\frac{1}{2}}$	$\xi = \mathbf{1}$	$\xi = \mathbf{2}$	$\xi = \mathbf{3}$	$\xi = \mathbf{4}$	$\xi = \mathbf{5}$	$\xi = \mathbf{6}$	$\xi = \mathbf{7}$
Frequentistische Modellierung									
grob	0.732	0.729	0.788	0.835	0.827	**0.860**	0.854	0.845	0.856
medium	0.847	0.845	0.839	**0.877**	0.861	0.856	0.861	0.873	0.860
fein	0.817	0.834	0.820	0.822	0.842	0.847	0.838	0.841	**0.850**
Bayes'sche Modellierung									
grob	0.731	0.733	0.713	0.847	0.832	0.852	0.857	0.850	**0.863**
medium	**0.842**	0.832	0.830	0.828	0.816	0.789	0.792	0.794	0.804
fein	0.819	0.793	0.816	**0.855**	0.833	0.851	0.854	0.853	0.853

11.3.3 Interpretation

Im Gegensatz zum Bogenschießen-Benchmark-Problem wurde hier die Anzahl der Beobachtungen konstant gehalten und die Diskretisierung verändert. Je feiner die Diskretisierung, desto größer ist die lokale Unsicherheit. Daher tendiert die Position des Maximums bzgl. ξ dazu, für eine fallende Anzahl an Zuständen anzusteigen. Bei der Brennkammer-Simulation (siehe Tab. 11.4) ist die Performance für die grobe Diskretisierung am höchsten, bei der Emissionsreduktions-Simulation (siehe Tab. 11.5) hingegen für die mittlere Diskretisierung.

In beiden Fällen führt die frequentistische Betrachtung in der Tendenz zu besseren Ergebnissen als unter Anwendung des uniformen Priors. Für die Brennkammer-Simulation kann die insgesamt beste Performance bei der groben Diskretisierung und dem frequentistischen Modell mit $\xi = 7$ erzielt werden, jedoch führt die Anwendung des Maximal-Entropie-Priors zu vergleichbaren Ergebnissen bereits bei $\xi = 3$. Für die Emissionsreduktions-Simulation wird das insgesamt beste Ergebnis bei der mittleren Diskretisierung und ebenfalls dem frequentistischen Modell mit $\xi = 2$ erreicht. Vergleichbare Ergebnisse können auch mit dem Maximal-Entropie-Prior erzielt werden. Im Gegensatz zum Bogenschießen-Benchmark bleiben die Performances auch bei Werten bis $\xi = 6$ stabil.

Das theoretische Optimum ist in beiden Fällen nicht bekannt, jedoch wurde für die Brennkammer-Simulation eine Performance von mindestens 0.861 unter Verwendung von 100000 Beobachtungen mit dem RPGNRR erzielt, im Falle der Emissionsreduktions-Simulation eine Performance von 0.61 unter den gleichen Bedingungen (siehe Abschn. 8.2). Während dort auf Grund der Einschränkungen an die Diskretisierung kein konkurrenzfähiges Ergebnis erzielt werden kann, erreicht die UPRL unter den gleichen Bedingungen auf der Brennkammer-Simulation im Einzelfall eine Performance von 0.863, was das Ergebnis der RPGNRR sogar übersteigt (siehe Tab. 11.6). Für die Brennkammer-Simulation sind die Ergebnisse der UPRL daher durchaus mit den besten bekannten Ergebnissen vergleichbar.

11.4 Qualitätssicherung bei der Steuerung von Gasturbinen

Den Nachweis der Qualitätssicherung in dem Sinne, dass die garantierte Minimalperformance gesteigert werden kann, an praktischen Beispielen zu erbringen, ist per Konstruktion nicht

möglich, da die Realität nur durch ein bestimmtes Modell repräsentiert wird und keine echte Verteilung über Modelle existiert. Eine Demonstration dieses Funktionsprinzips kann dennoch in begrenztem Maße durch eine Wahrscheinlichkeitsinversion erfolgen. Seien dazu M das Modell der Wirklichkeit, X die Beobachtungen und $Q(\pi, M)$ die Performance, die mit der Policy π auf dem Modell M erzielt werden kann. Während die UPRL die Verteilung $P(M|X)$ betrachtet, die schließlich zu einer Verteilung $P(Q(\pi(X), M)|X)$ für festes X führt, gewinnen wir aus den oben aufgeführten Experimenten die inverse Verteilung $P(X|M)$ und damit $P(Q(\pi(X), M)|M)$ für festes M.

Nun besteht die Möglichkeit der begrenzten Übertragbarkeit von $P(Q(\pi(X), M)|X)$ auf $P(Q(\pi(X), M)|M)$, denn mit einer sinkenden Unsicherheit bzgl. verschiedener Modelle für einen gegebenen Datensatz X muss im Durchschnitt auch die Unsicherheit bzgl. verschiedener Datensätze für ein gegebenes Modell M sinken. Für den Einzelfall kann für $P(Q(\pi(X), M)|X)$ zwar grundsätzlich keine Aussage getroffen werden, die Ergebnisse können aber durchaus eine Tendenz aufzeigen.

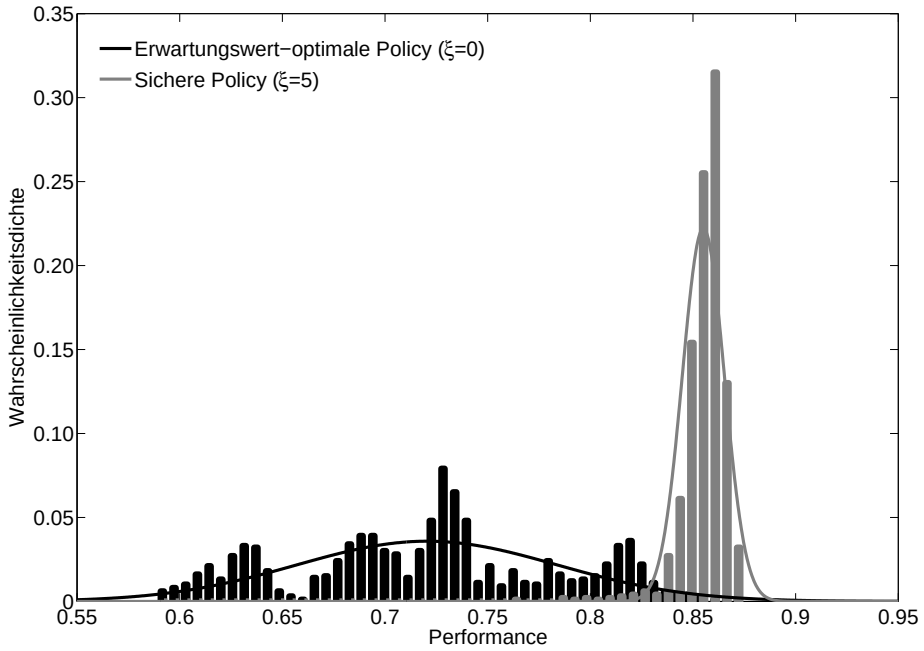


Abbildung 11.5: Performance-Verteilung für die Brennkammer-Simulation bzgl. unterschiedlicher Policies, die aus verschiedenen Datensätzen resultieren. Die Grafik zeigt die Histogramme für insgesamt jeweils 1000 Policies für die grobe Diskretisierung mit Bayes'scher Modellierung der Transitionswahrscheinlichkeiten mit uniformem Prior unter Berücksichtigung von 10000 Beobachtungen für $\xi = 0$ (schwarz) und $\xi = 5$ (grau). Durch den Randeffekt und die asymmetrische Exploration kommt es bei dieser Problemstellung mit steigendem ξ nicht nur zu einer verringerten Varianz, sondern auch zu einem Ansteigen des Erwartungswertes.

11.4.1 Ergebnisse

Diese inverse Verteilung ist in Abb. 11.5 exemplarisch für die Brennkammer-Simulation auf grober Diskretisierung mit uniformem Prior und 10000 Beobachtungen für $\xi = 0$ und $\xi = 5$ gezeigt. Das Histogramm bezieht sich also nicht auf verschiedene Modelle, sondern verschiedene Datensätze, die zur Bestimmung sicherheitsoptimaler Policies herangezogen wurden. Die Darstellung zeigt deutlich sowohl den Anstieg der Performance als auch die

Tabelle 11.7: Standardabweichung der Performance der Policies für die Brennkammer-Gasturbinen-Simulation mit frequentistischer und Bayes'scher Modellierung der Transitionswahrscheinlichkeiten mit uniformem Prior unter Berücksichtigung von jeweils 10000 Beobachtungen.

Diskr.	$\xi = \mathbf{0}$	$\xi = \frac{\mathbf{1}}{\mathbf{2}}$	$\xi = \mathbf{1}$	$\xi = \mathbf{2}$	$\xi = \mathbf{3}$	$\xi = \mathbf{4}$	$\xi = \mathbf{5}$	$\xi = \mathbf{6}$	$\xi = \mathbf{7}$
Frequentistische Modellierung									
grob	0.070	0.070	0.072	0.045	0.032	0.026	0.017	0.011	0.011
medium	0.061	0.057	0.059	0.043	0.036	0.036	0.039	0.033	0.032
fein	0.050	0.042	0.039	0.030	0.029	0.027	0.026	0.023	0.026
Bayes'sche Modellierung									
grob	0.061	0.066	0.048	0.022	0.018	0.013	0.011	0.016	0.017
medium	0.038	0.038	0.042	0.038	0.036	0.035	0.032	0.036	0.031
fein	0.060	0.053	0.045	0.042	0.037	0.040	0.038	0.044	0.048

Tabelle 11.8: Standardabweichung des Rewards für die Emissionsreduktions-Simulation mit frequentistischer und Bayes'scher Modellierung der Transitionswahrscheinlichkeiten mit uniformem Prior unter Berücksichtigung von jeweils 10000 Beobachtungen.

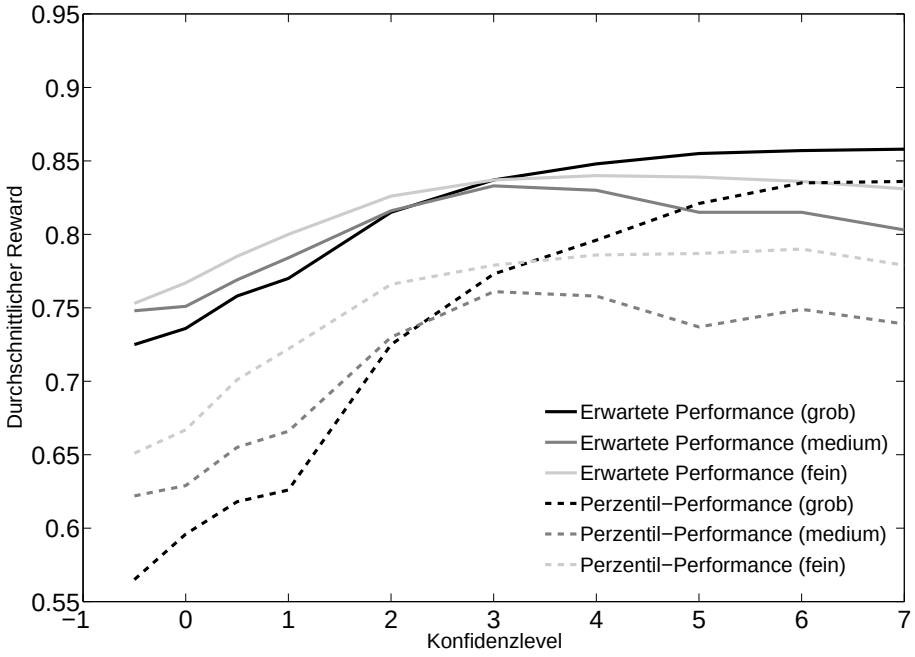
Diskr.	$\xi = \mathbf{0}$	$\xi = \frac{\mathbf{1}}{\mathbf{2}}$	$\xi = \mathbf{1}$	$\xi = \mathbf{2}$	$\xi = \mathbf{3}$	$\xi = \mathbf{4}$	$\xi = \mathbf{5}$	$\xi = \mathbf{6}$
Frequentistische Modellierung								
grob	0.199	0.196	0.190	0.165	0.169	0.167	0.169	0.175
medium	0.163	0.174	0.190	0.195	0.142	0.108	0.111	0.111
fein	0.092	0.173	0.172	0.242				
Bayes'sche Modellierung								
grob	0.183	0.180	0.155	0.120	0.092	0.089	0.080	0.084
medium	0.119	0.121	0.111	0.121	0.108	0.104	0.106	0.097

Verringerung des Risikos, dass durch einen ungünstig gewählten Datensatz die Performance zu gering ist. Damit ist experimentell verifiziert, dass diese Transformation möglich ist.

In den Tabn. 11.7 und 11.8 sind im Einzelnen die Standardabweichungen der Performance-Ergebnisse auf der Brennkammer-Simulation aufgeführt, die aus der Tatsache resultieren, dass den verschiedenen Policies unterschiedliche Datensätze zu Grunde liegen. In der Tendenz ist eine Stabilisierung der Performance mit steigendem ξ erkennbar. Bei hohen Werten für ξ steigt die Standardabweichung allerdings wieder, da der Grad an Stochastizität zunimmt. In der Abb. 11.6 sind die erwartete und die Minimalperformance für $\xi = 2$ für die Brennkammer-Simulation mit den unterschiedlichen Diskretisierungsstufen dargestellt. Man erkennt die Steigerung der Minimalperformance bis zu einem bestimmten Punkt für wachsendes ξ, wodurch eine Verbesserung der Sicherheit demonstriert werden kann. Es steigt also nicht nur der Erwartungswert, sondern es fällt auch das Risiko, eine schlechte Policy zu bestimmen.

11.4.2 Interpretation

Die Ergebnisse zeigen deutlich, dass die beiden Ziele, die die Unsicherheitspropagation auf dieser Problemklasse anstrebt, auch erreicht werden können. Der Randeffekt kann soweit nivelliert werden, dass die durchschnittliche Performance tatsächlich ansteigt. Darüber hinaus wird durch die parallel angestrebte Sicherheitsoptimalität auch die Varianz der Performances für Policies, die auf verschiedenen Datensätzen trainiert wurden, minimiert, so dass auch im Einzelfall, also für eine bestimmte beobachtete Trajektorie, eine Minimalperformance garantiert werden kann. Auch wenn dieser Effekt auf der inversen Verteilung die Sicherheitsoptimalität bzgl. der ursprünglichen Verteilung über mögliche MDP nicht nachweisen kann, ist es jedoch ein wertvolles Indiz dafür und für den praktischen Einsatz von großer Bedeutung.



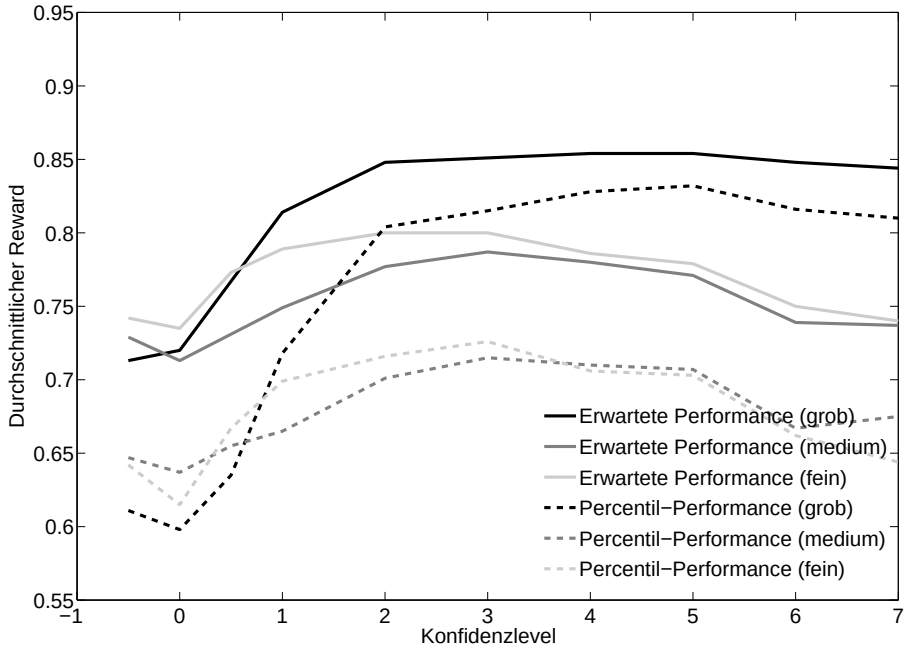


Abbildung 11.6: Erwartete und inverse Perzentil-Performance für die Brennkammer-Simulation mit frequentistischer (oben) und Bayes'scher Modellierung der Transitionswahrscheinlichkeiten (unten). Die Perzentil-Performance entspricht dabei dem Konfidenzlevel $\xi = 2$. Wie deutlich zu sehen ist, steigt für die frequentistische Modellierung nicht nur die Performance, sondern auch die Perzentil-Performance bis $\xi = 3$ an und bleibt darüber stabil. Auch für die Bayes'sche Modellierung erkennt man das zunächst wachsende Verhalten von Performance und Perzentil-Performance, die nach $\xi = 3$ wieder leicht fallen.

Teil IV

Schlussfolgerungen

Kapitel 12

Schlussfolgerungen und Ausblick

In der vorliegenden Arbeit haben wir die Philosophien sowohl der strukturellen Risikominimierung als auch des Bayes'schen Paradigmas mit Reinforcement-Learning kombiniert und sind damit einen prinzipiellen Weg gegangen, die Generalisierungsleistung und damit die Informationseffizienz zu steigern. Dieses Ziel haben wir erreichen und an Benchmarks und industriellen Anwendungen praktisch nachweisen können.

12.1 Beiträge

Die Beiträge der vorliegenden Arbeit bestehen im Einzelnen in den folgenden Komponenten.

1. Steigerung der Informationseffizienz durch strukturelle Risiko-Minimierung
   (a) Definition der Rewards-Regression als Verallgemeinerung der Fitted-Q-Iteration
   (b) Direkte Umsetzung des Konzeptes der strukturellen Risiko-Minimierung durch Anwendung von Kernel-Machines für die Rewards-Regression
      i. Nachweis der Identität der Optimalitätskriterien Fixpunkt der Bellman-Iteration und Minimum des Bellman-Residuums
      ii. Direkte Policy-Identifikation mittels quadratischer Programmierung zur Vermeidung von Policy-Iteration
      iii. Ansätze für rekurrente Support-Vector-Machines für partiell beobachtbare Probleme
   (c) Anwendung von neuronalen Netzen für die Rewards-Regression
      i. Betrachtung verschiedener Optimalitätskriterien und Iterationstechniken
      ii. Kombination mit Policy-Gradient-Verfahren für kontinuierliche Aktionsräume und zur weiteren Steigerung der Informationseffizienz
      iii. Kombination mit rekurrenten neuronalen Netzen für partiell beobachtbare Systeme
   (d) Theoretische Aussagen zur Rewards-Regression
      i. Bestimmung einer Performance-Schranke in Abhängigkeit von der Approximationsgüte der Rewards-Regression
      ii. Nachweis der Nicht-Existenz eines erwartungstreuen Schätzers für das Bellman-Residuum im Batch-Learning-Fall
      iii. Nachweis der Rekonstruierbarkeit von Markov-Zuständen in partiell beobachtbaren Systemen
   (e) Nachweis der Steigerung der Informationseffizienz auf Benchmarks und Gasturbinen-Simulationen

2. Qualitätssicherung und Steigerung der Informationseffizienz durch Unsicherheitspropagation
   (a) Anwendung der Unsicherheitspropagation auf die Bellman-Iteration
      i. Unsicherheitspropagation im diskreten Fall zur Bestimmung der garantierten Performance mit einer bestimmten Wahrscheinlichkeit
      ii. Nachweis von Existenz und Eindeutigkeit der Lösung
      iii. Demonstration der Anwendbarkeit verschiedener statistischer Paradigmen und unterschiedlicher Funktionsapproximatoren für eine breite Anwendbarkeit
   (b) Konzept der Sicherheitsoptimalität
      i. Verfahren zur Optimierung der garantierten Performance
      ii. Berücksichtigung stochastischer sicherheitsoptimaler Policies für die Gewährleistung von Diversifikation
      iii. Optimierung der vollständigen Kovarianzmatrix zur Betrachtung von globaler Sicherheitsoptimalität im Gegensatz zur lokalen Sicherheitsoptimalität
      iv. Ansätze zur Berücksichtigung höherer Momente der Verteilung der Performance für eine breitere Anwendbarkeit
   (c) Praktischer Nachweis der Sicherheitsoptimalität
      i. Nachweis der Funktionsweise der Unsicherheitspropagation auf der Bellman-Iteration zur Gewährleistung der Qualitätssicherung auf künstlichen Benchmarks
      ii. Nachweis der Steigerung der Informationseffizienz auf künstlichen Benchmarks und Gasturbinen-Simulationen
      iii. Ansätze zum Nachweis der Qualitätssicherung auf Gasturbinen-Simulationen

12.2 Gegenüberstellung der Verfahren und Anwendungsempfehlungen

Wir haben mehrere Varianten der Rewards-Regression vorgestellt, neben der Anwendung auf Kernel-Machines und neuronalen Netzen wurden auch Erweiterungen zur Beherrschung allgemeinerer Problemklassen berücksichtigt. Die Kernel-Rewards-Regression ist, zusammen mit lokalen Kernels, dann zu bevorzugen, wenn die Datendichte der Beobachtungen im Zustands-Aktions-Raum stark variiert. Die Lokalität des Funktionsapproximators sorgt dann für eine gleichmäßig gute Lösung. Wenn Vorwissen über die Transitionswahrscheinlichkeiten oder die Q-Funktion verfügbar ist, so dass ein angemessener Kernel konstruiert werden kann, ist es ebenfalls zweckmäßig, die KRR anzuwenden.

Die EKRR hat insbesondere den Vorteil, dass ohne Policy-Iteration eine Lösung ermittelt werden kann. Das Bestimmen der optimalen Policy erfolgt in einem geschlossenen quadratischen Programm. KRR mit Policy-Iteration und einer stochastischen Komponente kann das Finden einer optimalen Policy im starken Sinne ebenfalls garantieren. Die Iteration über stochastische Policies erlaubt darüber hinaus, eine schwächere Form von Optimalität zu erreichen, wenn keine stark optimale Policy existiert. Die Wahl des Kernels ermöglicht in begrenztem Umfang auch den Einsatz kontinuierlicher Aktionen und, als Intrinsic-Recurrent-Support-Vector-Machine, die Beherrschbarkeit von POMDP.

Die Neural-Rewards-Regression bietet die Vorteile globaler Funktionsapproximatoren, die i.d.R. mit weniger Parametern auskommen und einen im Gegensatz zu den lokalen Verfahren umfassenderen Überblick über die Problemstellung gewinnen können. Durch das im Vergleich zur konvexen Optimierung allgemeinere Optimierungsverfahren des Gradientenabstiegs konnten wir die NRR darüber hinaus in der Praxis für eine wesentlich breitere Problemklasse zur Anwendung bringen. Die unterschiedlichen Varianten der NRR erlauben au-

ßerdem die Beherrschbarkeit verschiedener Optimalitätskriterien sowie beliebige ihrer Kombinationen. Dies betrifft diskrete und kontinuierliche Aktionsräume und MDP höherer Ordnung sowie die Kriterien Fixpunkt der Bellman-Iteration, Minimum des Bellman-Residuums und des Auxiliared-Bellman-Residuums.

Die Unsicherheitspropagation für Reinforcement-Learning bietet demgegenüber einen zusätzlichen Mehrwert. Durch die Bestimmung und Maximierung einer mit einer bestimmten Wahrscheinlichkeit garantierten Minimalperformance können Anforderungen an die Qualitätssicherung erfüllt, aber auch weitere Anwendungsfälle wie Wettkampfsituationen oder informationssuchende Exploration erschlossen werden.

Wie an den Ergebnissen auf den Gasturbinen-Simulationen zu erkennen ist, sind die Rewards-Regression und die Unsicherheitspropagation im Hinblick auf ihre Performance in dieser Problemklasse durchaus vergleichbare Ansätze. Erwartungsgemäß liegt die Performance der Neural-Rewards-Regression auf der Emissionsreduktions-Simulation deutlich und bei der Brennkammer-Simulation noch leicht über der der Unsicherheitspropagation für deterministische Zustands- und Aktionsräume, im Einzelfall kann jedoch für 100000 Beobachtungen mit der UPRL und der Bayes'schen Modellierung eine höhere Performance erzielt werden.

Die Stärken und Vorzüge der UPRL sind damit jedoch noch nicht ausgereizt, da eine praktische Anwendung auf Funktionsapproximatoren noch aussteht. Wegen der aber bereits sehr hohen Performance auf der Brennkammer-Simulation besteht auch die Möglichkeit, sich die positiven Eigenschaften der diskreten Verfahren, die vor allem in der besseren Beherrschbarkeit und Stabilität sowie in der kürzeren Laufzeit bestehen, zu Nutze zu machen und die UPRL als zur Rewards-Regression orthogonalem Verfahren zum Einsatz zu bringen.

Die UPRL kann daher zur Optimierung der Minimalperformance bzgl. der Modellwahl, zur Stabilisierung der Performance bzgl. der Datenauswahl und zur Steigerung der Performance in einer wichtigen Problemklasse eingesetzt werden. Darüber hinaus ist das Verfahren in der erprobten Form stabil, schnell und insgesamt konkurrenzfähig mit anderen dateneffizienten Methoden.

12.3 Ausblick

Für die Zukunft gilt es zu beiden Schwerpunkten noch viele offene Fragen zu beantworten. Zum theoretischen Verständnis der Generalisierungsfähigkeit fehlt nach wie vor ein umfassendes Konzept an Kapazitätsmaßen und Performanceschranken, die auf Kapazitätsaussagen beruhen. Darüber hinaus ist es zielführend, sich weiter mit der Frage der Steigerung der Informationseffizienz bei verschränkter Regularisierung von Q-Funktion und Policy zu beschäftigen und diese theoretisch zu durchdringen.

Ein wesentlicher offener Punkt zur Anwendung der Unsicherheitspropagation besteht in der Berücksichtigung von Gliedern höherer Ordnung der Taylorentwicklung. Insbesondere beim diskreten Ansatz und der LSPI genügen bereits die ersten beiden Taylor-Glieder. Außerdem wird die Entwicklung einer algorithmischen Anwendung der Berücksichtigung höherer statistischer Momente angestrebt.

Am wichtigsten ist jedoch die weitere industrielle Erprobung der hier vorgestellten Ansätze. Dabei wird eine Ausweitung auf Anwendungen im Bereich der Windturbinen-Steuerung und im Finanzdienstleistungsbereich angestrebt. Die in der vorliegenden Arbeit vorgestellten Verfahren, ihre Orthogonalität sowie unterschiedlichen Eigenschaften und Beiträge stellen auch wesentliche Schritte hin zu echtem autonomen Lernen dar, also zu Methoden, die selbstständig, dateneffizient, sicher, unsicherheitsberücksichtigend die Umwelt explorieren, Wissen generieren und dieses schließlich nutz- und gewinnbringend einsetzen.

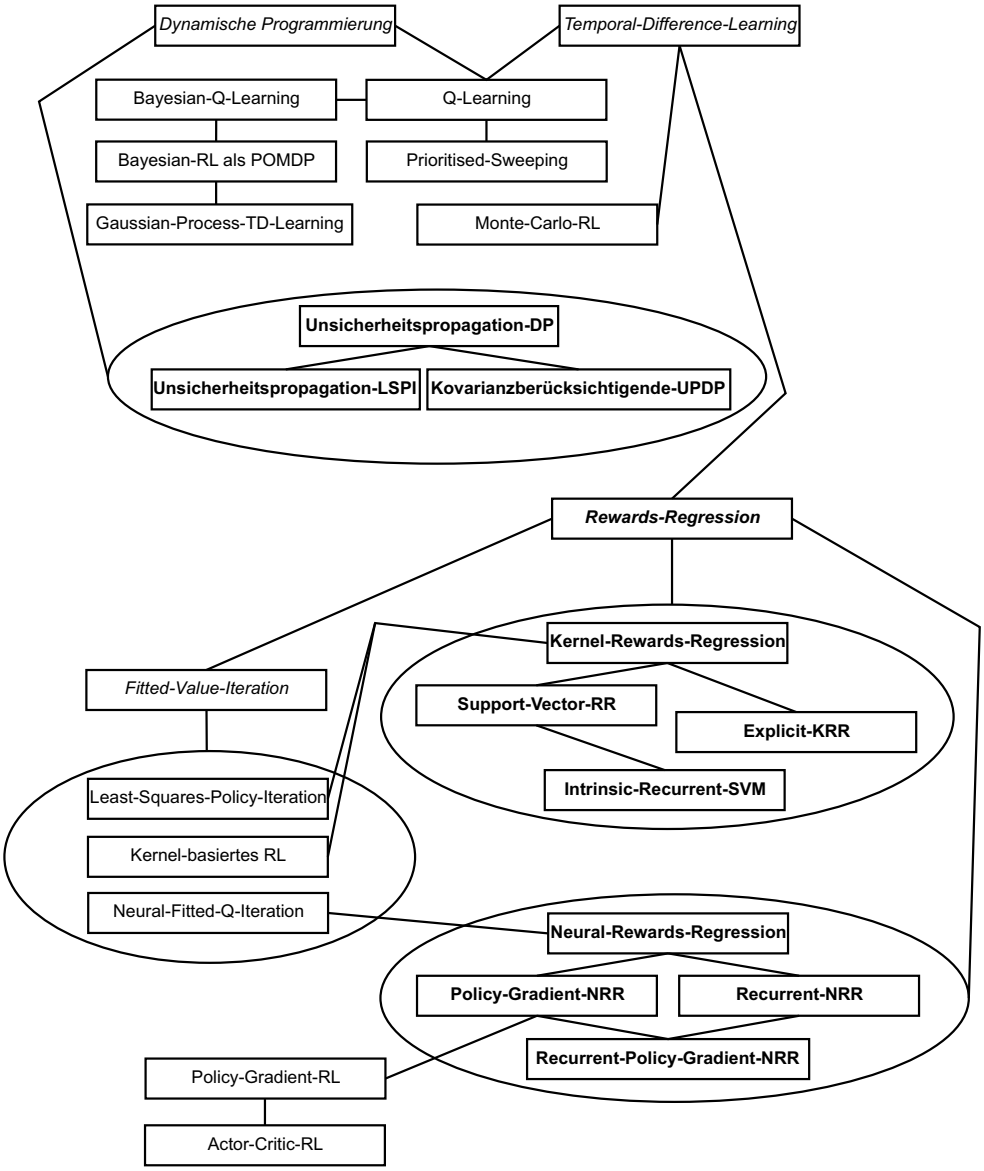


Abbildung 12.1: Illustration der Zusammenhänge und Verbindungen zwischen den etablierten (in normaler Schrift) und den in der vorliegenden Arbeit (fettgedruckt) vorgestellten Methoden. Die Verbindungen beziehen sich auf inhaltliche Abhängigkeiten zwischen den verschiedenen Verfahren. Mehrere Verfahren, die eingekreist sind, bilden eine Einheit, die gegebenenfalls einer Verfahrensgruppe (Rewards-Regression oder Fitted-Value-Iteration) untergeordnet sind.

Anhang

Glossar

A posteriori Verteilung beschreibt die angenommene Wahrscheinlichkeit für das Auftreten verschiedener Hypothesen, nachdem die Kenntnisse über die Hypothesen durch Beobachtungen erhärtet wurden.

A priori Verteilung beschreibt die ohne Beobachtungen angenommene Wahrscheinlichkeit für das Auftreten verschiedener Hypothesen.

Actor-Critic-Methoden kombinieren bewertungsfunktions-basierte Methoden mit Policy-Gradient-Methoden.

Arbeitspunkt ist ein Systemzustand eines dynamischen Systems, der als erstrebenswert gilt oder in dem das System gehalten werden soll.

Auxiliared-Bellman-Residuum ist eine erweiterte Form des Bellman-Residuums, das die Erwartungstreue verbessert.

Back-Propagation-Verfahren ist eine Klasse von Algorithmen zur effizienten Realisierung des Gradientenabstiegsverfahrens für neuronale Netze.

Batch-Learning besteht in der Anwendung von Machine-Learning-Verfahren und der Generierung von Hypothesen basierend auf einem zuvor festgelegten Datensatz, etwa einer Trajektorie.

Bayesian-Q-Learning kombiniert Q-Learning mit Bayes'schen Methoden zur Bestimmung einer Unsicherheit der Q-Funktion.

Bayesian-Reinforcement-Learning als POMDP besteht in der Zurückführung auf partiell beobachtbare Markov-Entscheidungsprozesse. Die unbekannten Parameter werden dabei zum Teil des Zustandsraumes, der nicht beobachtet werden kann.

Bayes'sche Inferenz besteht im Schließen von einer a priori Verteilung auf eine a posteriori Verteilung durch Berücksichtigen beobachteter Daten, etwa Trajektorien.

Bellman-Iteration ist über die Bewertungsfunktionen oder Q-Funktionen definiert. Sie konvergiert gegen eine Lösung der Bellman-Gleichung oder der Bellman-Optimalitäts-Gleichung, falls der Bellman-Operator eine Nicht-Expansion ist.

Bellman-Gleichung ist eine Fixpunkt-Gleichung über den Raum möglicher Bewertungsfunktionen oder Q-Funktionen. Ein Fixpunkt ist durch eine Bewertungsfunktion oder Q-Funktion gegeben, die den erwarteten Gesamtertrag der angewendeten Policy zurückgibt.

Bellman-Operator wirkt auf Funktionen vom Typ Bewertungsfunktion oder Q-Funktion. Mehrmaliges Anwenden führt zur Bellman-Iteration.

Bellman-Optimalitäts-Gleichung ist eine Fixpunkt-Gleichung über den Raum möglicher Bewertungsfunktionen oder Q-Funktionen. Ein Fixpunkt ist durch eine Bewertungsfunktion oder Q-Funktion gegeben, die den erwarteten Gesamtertrag der optimalen Policy zurückgibt.

Bellman-Residuum ist, bezüglich einer geeignet gewählten Norm, der Abstand einer Bewertungsfunktion oder Q-Funktion zu der Funktion, die sich durch einmalige Anwendung des Bellman-Operators ergibt.

Bewertungsfunktion ist ein Schätzer für den Gesamtertrag ausgehend von einem bestimmten Zustand.

Bias-Variance-Dilemma besteht in dem Problem, Bias und Varianz eines Schätzers in optimaler Weise gegeneinander abzustimmen. Durch Vergrößern der Kapazität des Hypothesenraumes wird i.d.R. der Bias reduziert und die Varianz vergrößert und umgekehrt.

Konjugierter Prior ist eine a priori Verteilung zu einer Likelihood-Wahrscheinlichkeitsverteilung, so dass jede a posteriori Verteilung zur gleichen Funktionsklasse gehört.

Deterministische Policy ist eine Abbildung vom Zustands- in den Aktionsraum und wird als Aktionsauswahlregel angewendet. Im jeweiligen Zustand wird die zurückgegebene Aktion ausgeführt.

Diversifikation besteht in der Verteilung von Entscheidungen auf mehrere Alternativen. I.e.S. verstehen wir unter Diversifikation die Bestimmung stochastischer Policies zur Minimierung der Unsicherheit ihrer Performance.

Dynamische Systeme sind mathematische Modelle mit einer zeitlichen Abhängigkeit. Viele dynamische Systeme können als Markov-Entscheidungsprozess beschrieben werden.

Entropie ist der Erwartungswert des Logarithmus der zu Grunde liegenden Zufallsvariable und damit eine Kenngröße einer Verteilung. Die Entropie ist umso kleiner, je mehr Information in der Verteilung enthalten ist.

Ergodizität eines Markov-Prozesses oder eines Markov-Entscheidungsprozesses mit einer bestimmten Policy liegt vor, sofern für jedes Paar von Zuständen die Wahrscheinlichkeit für das Erreichen des einen Zustandes ausgehend vom anderen Zustand nach endlich vielen Schritten positiv ist.

Erwartungstreue eines Schätzers liegt dann vor, wenn der Erwartungswert des durch den Schätzer bestimmten Schätzwertes bezüglich der Menge aller möglichen Realisierungen dem wahren Wert entspricht.

Explorations-Ausbeutungs-Dilemma besteht in dem Problem, Exploration und Ausbeutung (Exploitation) gegeneinander abzustimmen. Je größer die Explorationskomponente, desto schneller gewinnt man Informationen über das System, aber desto später kann man durch Anwenden einer optimalen Policy von einem hohen Gesamtertrag profitieren.

Fast sichere Eigenschaften treffen mit Wahrscheinlichkeit 1 zu. Im Gegensatz zu sicheren Eigenschaften können sie jedoch eventuell auch nicht zutreffen, höchstens jedoch auf einer Menge mit Maß 0.

Fitted-Value-Iteration ist eine spezielle Form der Bellman-Iteration, bei der sukzessive Funktionsapproximatoren für die Bewertungsfunktion bestimmt werden.

Gaussian-Process-Temporal-Difference-Learning ist ein Reinforcement-Learning-Verfahren, das die Funktionswerte der Bewertungsfunktionen auf den Beobachtungen als Gauß-Prozess auffasst und dadurch die Bewertungsfunktion bestimmt.

Generalisierung ist der Vorgang des Schließens vom Speziellen auf das Allgemeine durch ein Machine-Learning-Verfahren.

Gesamtertrag ist die diskontierte Summe aller Rewards, die bei der Anwendung einer bestimmten Policy ausgehend von einem bestimmten Folgezustand erzielt wird.

Globale Sicherheitsoptimalität ist Sicherheitsoptimalität auf globaler Ebene, also gemeinsam für alle Zustands-Aktions-Paare unter Berücksichtigung der Kovarianzen zwischen den Zustands-Aktions-Paaren.

Hypothesenraum ist eine Menge von Funktionen, die einen Zusammenhang einer bestimmten Art einer Beschreibung der Realität abbilden können.

Induktion ist der Vorgang des Schließens vom Speziellen auf das Allgemeine. Dabei wird aus Beobachtungen eine allgemeine Hypothese über den zu Grunde liegenden Zusammenhang aufgestellt.

Jacobi-Matrix beschreibt die partiellen Ableitungen einer multivariaten Funktion von ihren multivariaten Argumenten.

Kapazität einer Menge von Funktionen quantisiert die Darstellbarkeit funktionaler Zusammenhänge durch die Wahl von Funktionen aus der entsprechenden Menge. Wichtige Kapazitätsmaße sind die VC-Dimension und die minimale Beschreibungslänge.

Kernel-Machines sind Funktionsapproximatoren, die sich Kernels zu Nutze machen. Ein Kernel beschreibt dabei das Skalarprodukt in einem Feature-Raum.

Kernel-Rewards-Regression ist eine Klasse von Reinforcement-Learning-Verfahren, die Ausprägungen des Rewards-Regression-Ansatzes sind und auf Kernel-Machines beruhen.

Kernel-basiertes Reinforcement-Learning ist ein Fitted-Q-Iteration-Verfahren. Die Funktionsapproximatoren verwenden Kernel-Funktionen zusammen mit einer Erwartungswertschätzung.

Konsistenz eines Schätzers liegt dann vor, wenn der Schätzwert des Schätzers mit gegen unendlich gehender Anzahl an Beobachtungen gegen den wahren zu schätzenden Wert konvergiert.

Konvexe Programmierung wird verwendet zur Lösung konvexer Probleme. Dazu zählen insbesondere die lineare und die quadratische Programmierung, die bei vielen Problemstellungen im Machine-Learning Anwendung finden.

Konzeptklasse ist eine Menge von Funktionen, die einen Zusammenhang einer bestimmten Art einer Realität abbilden können.

Kovarianzmatrix beschreibt die statistischen Kovarianzen zwischen den Schätzern verschiedener zu schätzender Werte.

Least-Squares-Policy-Iteration ist ein Fitted-Q-Iteration-Verfahren. Die Funktionsapproximatoren verwenden lineare Funktionen zusammen mit Least-Squares-Verfahren.

Lokale Sicherheitsoptimalität ist Sicherheitsoptimalität auf lokaler Ebene, also für jedes mögliche Zustands-Aktions-Paar.

Markov-Entscheidungsprozesse sind stochastische Prozesse über Zustände, in denen Aktionen ausgeführt werden können und für die die Wahrscheinlichkeit für das Auftreten eines jeweiligen Zustandes nur von Vorgängerzustand und -aktion abhängt.

Markov-Entscheidungsprozesse höherer Ordnung sind stochastische Prozesse über Zustände, in denen Aktionen ausgeführt werden können und für die die Wahrscheinlichkeit für das Auftreten eines jeweiligen Zustandes höchstens von einer bestimmten Anzahl von Vorgängerzuständen und -aktionen abhängt.

Markov-Prozesse sind stochastische Prozesse über Zustände, für die die Wahrscheinlichkeit für das Auftreten eines jeweiligen Zustandes nur vom Vorgängerzustand abhängt.

Markov-Zustände enthalten alle nötigen Informationen, um, zusammen mit einer eventuell auswählbaren Aktion, die vollständige Wahrscheinlichkeitsverteilung über die möglichen Folgezustände bestimmen zu können.

Maximum-Margin ist ein Optimalitätskriterium, das bei Support-Vector-Machines verwendet wird. Demnach ist Optimalität eines Klassifikationsproblems dann erreicht, wenn (innerhalb des Feature-Raumes) der Abstand der am dichtesten an der die Klassen trennenden Hyperebene liegenden Vektoren maximal ist.

Minimale Beschreibungslänge ist ein Kapazitätsmaß, das die Menge an Information bezeichnet, die erforderlich ist, um eine Hypothese aus dem Hypothesenraum zu beschreiben.

Modellbasiertes Reinforcement-Learning ist Reinforcement-Learning unter Berücksichtigung eines expliziten Modells der Dynamik, also der Transitionswahrscheinlichkeiten. Dieses Modell kann unter Zuhilfenahme von Vorwissen und auf der Grundlage beobachteter Daten bestimmt werden.

Modellfreies Reinforcement-Learning ist Reinforcement-Learning ohne Berücksichtigung eines expliziten Modells der Dynamik, also der Transitionswahrscheinlichkeiten. Die optimale Policy wird direkt unter Berücksichtigung der Daten bestimmt.

Monte-Carlo-Methoden ist eine Klasse von Reinforcement-Learning-Verfahren, die Bewertungsfunktionen oder Q-Funktionen optimieren, indem sie die wahre Bewertungsfunktion oder Q-Funktion durch Monte-Carlo-Simulationen approximieren.

Neural-Rewards-Regression ist eine Klasse von Reinforcement-Learning-Verfahren, die Ausprägungen des Rewards-Regression-Ansatzes sind und auf neuronalen Netzen beruhen.

Neural-Fitted-Q-Iteration ist ein Fitted-Q-Iteration-Verfahren. Die Funktionsapproximatoren verwenden neuronale Netze zusammen mit dem Back-Propagation-Verfahren.

Neuronales Netz ist ein Funktionsapproximator, dessen Funktionsprinzip an natürliche Nervensysteme angelehnt ist. Neuronale Netze bestehen aus Neuronen, die durch Synapsen miteinander verbunden sind.

Nicht-Expansion ist ein Operator, nach deren Anwendung auf zwei unterschiedliche Argumente der Abstand zwischen den beiden Ergebnissen nicht größer ist als der Abstand zwischen den Argumenten.

Nicht-stationäre Markov-Entscheidungsprozesse sind Markov-Entscheidungsprozesse, für die sich die Transitionswahrscheinlichkeitsverteilung im Verlaufe der Zeit verändert.

Off-Policy-Learning ist eine Klasse von Reinforcement-Learning-Verfahren, die Policies optimieren, ohne dass die zur Exploration verwendete Policy selbst gegen die optimale Policy konvergieren muss.

On-Policy-Learning ist eine Klasse von Reinforcement-Learning-Verfahren, die Policies optimieren, für die die zur Exploration verwendete Policy selbst gegen die optimale Policy konvergieren muss oder, falls sie es nicht tut, im Allgemeinen nur eine suboptimale Policy erreicht werden kann.

Online-Learning besteht in der Anwendung von Machine-Learning-Verfahren und der Generierung von Hypothesen, während online Daten gesammelt werden, etwa eine Trajektorie gefahren wird.

Optimalregelungsprobleme werden als Probleme der optimalen Zustandsregelung bezeichnet und bestehen darin, eine optimale Aktionsauswahlregel (Policy) zur Regelung eines dynamischen Systems abzuleiten. Sie grenzen sich von Optimalsteuerungsproblemen ab, bei denen der Regler (die Policy) nicht auf das zu regelnde System rückkoppeln kann.

Partiell beobachtbare Markov-Entscheidungsprozesse sind Markov-Entscheidungsprozesse, bei denen der Zustandsraum nicht vollständig beobachtet werden kann.

Policy beschreibt eine Aktionsauswahlregel, i.d.R. zur Regelung von dynamischen Systemen.

Policy-Evaluation bestimmt die Bewertungsfunktion oder die Q-Funktion für eine gegebene Policy.

Policy-Iteration ist eine Klasse von Reinforcement-Learning-Verfahren, die Bewertungsfunktionen oder Q-Funktionen optimieren, indem sie auf einen zuvor beobachteten Datensatz zurückgreifen.

Policy-Gradient-Methoden verwenden keine explizite Darstellung der Bewertungsfunktion zur Bestimmung der optimalen Policy. Sie greifen stattdessen auf eine explizite Darstellung der Policy zurück.

Probably-Approximately-Correct-Learning sind Machine-Learning-Verfahren, für die die Wahrscheinlichkeit dafür, dass die ermittelte Hypothese zu weit vom wahren Konzept abweicht, in Abhängigkeit von der Anzahl der zur Verfügung stehenden Beobachtungen nach oben abgeschätzt werden kann.

Q-Funktion ist ein Schätzer für den Gesamtertrag ausgehend von einem bestimmten Zustand unter Anwendung einer bestimmten Aktion.

Qualitätssicherung ist ein Prozess zur Absicherung der Güte von Policies durch Berücksichtigen der Unsicherheit ihrer geschätzten Performance. Das Ziel besteht darin, keine schlechten Policies auszuliefern, die Qualität der Policies zu quantifizieren und die mit einer bestimmten Wahrscheinlichkeit zugesicherte Minimalperformance zu maximieren.

Randphänomen wird durch eine konservative asymmetrische Exploration des Zustandsraumes verursacht. Der Rand des explorierten Bereiches wird dabei selten besucht, so dass es dort zu einer erhöhten Unsicherheit für die Parameter des MDP kommt. Die dadurch im Einzelfall überschätzten Q-Werte an diesen Zuständen führen zu Policies minderer Qualität.

Regularisierung ist die systematische Einschränkung von Hypothesenräumen zur Reduktion ihrer Kapazität oder die systematische Umgewichtung von Hypothesen hinsichtlich zu berücksichtigender Vorannahmen oder prinzipieller Annahmen über die Konzeptklasse.

Reinforcement-Learning ist der Ansatz des Machine-Learnings zur Lösung von Optimalsteuerungs- und -regelungsproblemen. Das Prinzip besteht in der Herleitung von Handlungsstrategien (Policies) auf der Grundlage von beobachteten Explorations-Trajektorien.

Rekurrente neuronale Netze sind neuronale Netze, die zeitliches Verhalten und dynamische Systeme modellieren können. Sie werden für Prognose-, Steuerungs- und Regelungsaufgaben herangezogen.

Reward ist die Belohnung, die beim Übergang von einem Zustand durch Ausführen einer Aktion in einen bestimmten Folgezustand vereinnahmt wird.

Rewards-Regression ist eine Klasse von Reinforcement-Learning-Verfahren, bei dem das Reinforcement-Learning-Problem in ein geschlossenes Regressionsproblem überführt wird.

Ridge-Regression ist eine Regressionsmethode, die auf dem Least-Squares-Verfahren beruht. Die Methode forciert Lösungen mit kleinen Gewichten durch Einführung einer Regularisierungskomponente.

Shared-Weights ermöglichen verschiedene Gewichte eines neuronalen Netzes so miteinander zu verbinden, dass sie durch denselben Wert ausgedrückt werden. Diese Verknüpfung hat entsprechend Auswirkungen auf das Gradientenabstiegsverfahren.

Sicherheitsoptimalität ist eine Form der Optimalität in Optimalsteuerungs- und -regelungsproblemen, bei der nicht die erwartete Performance, sondern die Performance an einem bestimmten Perzentil maximiert wird.

Statistische Lerntheorie beschäftigt sich mit der mathematisch-statistischen Begründung und Fundierung des Machine-Learnings.

Statistische Momente sind Kenngrößen von Verteilungen und bestehen in der erwarteten potenzierten Abweichung zu einem bestimmten Referenzwert, etwa dem Erwartungswert der Verteilung. Erwartungswert, Varianz, Schiefe und Kurtosis sind statistische Momente.

Stochastische Policy ist eine Abbildung vom Zustands-Aktions-Raum in die reellen Zahlen und wird als Aktionsauswahlregel angewendet. Im jeweiligen Zustand wird die jeweilige Aktion mit der zurückgegebenen Wahrscheinlichkeit ausgeführt.

Strukturelle Risiko-Minimierung besteht in der Minimierung des Risikos, mit empirisch gewonnen Hypothesen aus einem bestimmten Hypothesenraum vom wahren Konzept zu weit abzuweichen. Dazu definiert man eine Struktur von Hypothesenräumen, durch Auswahl des optimalen Hypothesenraumes aus der Struktur wird das wahre Konzept durch die darin optimale Hypothese insgesamt am besten abgebildet.

Support-Vector-Machine ist ein Funktionsapproximator, dessen Funktionsprinzip der statistischen Lerntheorie entnommen ist und dem Prinzip des Maximum-Margins folgt. Support-Vector-Machines basieren auf Kernel-Machines.

Temporal-Difference-Learning ist eine Klasse von Reinforcement-Learning-Verfahren, die Bewertungsfunktionen oder Q-Funktionen optimieren, indem sie ihre temporale Differenz, also die Abweichung zu einem jeweiligen Schätzer minimieren.

Trajektorie ist eine zusammenhängende Folge von beobachteten Zuständen, Aktionen und vereinnahmten Rewards, die von einer bestimmten Policy oder Explorationsstrategie gewonnen wird.

Transitionswahrscheinlichkeit ist die Wahrscheinlichkeit für den Übergang von einem Zustand durch Auswahl einer Aktion in einen bestimmten Folgezustand.

Unsicherheit bezeichnet eine Größe, die mit einem Schätzer verknüpft werden kann und seine Streuung um den wahren Wert beschreibt.

Unsicherheitspropagation wird auch als Gauß'sche Methode der Fehlerfortpflanzung bezeichnet und überführt die Unsicherheit von Messwerten in die Unsicherheit von Funktionswerten, die sich durch einen funktionalen Zusammenhang aus den Messwerten ergeben.

Value-Iteration ist eine spezielle Form der Bellman-Iteration für diskrete Zustands- und Aktionsräume.

Vapnik-Chervonenkis-Dimension ist ein Kapazitätsmaß. Die VC-Dimension ist die Anzahl der Punkte, die bei günstigster Wahl in jeder möglichen Art indiziert werden können.

Literaturverzeichnis

[1] Achbany, Y. ; Fouss, F. ; Yen, L. ; Pirotte, A. ; Saerens, M. : Optimal tuning of continual online exploration in reinforcement learning. In: Kollias, S. (Hrsg.) ; Stafylopatis, A. (Hrsg.) ; Duch, W. (Hrsg.) ; Oja, E. (Hrsg.): *Proc. of the International Conference. Proceedings, Part I*, 2006, S. 790–800

[2] Anthony, M. ; Bartlett, P. L.: *Neural Network Learning Theoretical Foundations*. International Organization for Standardization, 1993

[3] Antos, A. ; Szepesvári, C. ; Munos, R. : Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. In: *Proc. of the Conference on Learning Theory*, 2006, S. 574–588

[4] Antos, A. ; Szepesvári, C. ; Munos, R. : Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. In: *Machine Learning* 71 (2008), Nr. 1, S. 89–129

[5] Appl, M. ; Brauer, W. : Fuzzy model-based reinforcement learning. In: *Advances in Computational Intelligence and Learning*, 2002, S. 211–223

[6] Baird, L. C.: Residual algorithms: Reinforcement learning with function approximation. In: *Proc. of the International Conference on Machine Learning*, 1995, S. 30–37

[7] Baird, L. C. ; Klopf, A. : Reinforcement learning with high-dimensional, continuous actions / Wright Laboratory, Wright-Patterson Air Force Base. 1993 (OH 45433-7301). – WL-TR-93-1147

[8] Banach, S. : *Théorie des opérations linéaires*, Universität Warschau, Diss., 1933

[9] Barto, A. G. ; Sutton, R. S. ; Anderson, C. W.: Neuronlike adaptive elements that can solve difficult learning control problems. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13 (1983), S. 834–846

[10] Bayes, T. : An essay towards solving a problem in the doctrine of chances, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S. In: *Philosophical Transactions* 53 (1763), S. 269–271

[11] Bellman, R. : *Dynamic Programming and Stochastic Control Processes*. Rand Corporation, 1957

[12] Bertsekas, D. P. ; Tsitsiklis, J. N.: *Parallel and Distributed Computing: Numerical Methods*. Prentice Hall, 1989

[13] Bertsekas, D. P. ; Tsitsiklis, J. N.: *Neuro-Dynamic Programming*. Athena Scientific, 1996

[14] Byrd, R. ; Gilbert, J. C. ; Nocedal, J. : A trust region method based on interior point techniques for nonlinear programming. In: *Mathematical Programming* 89 (2000), Nr. 1, S. 149–185

[15] Coppersmith, D. ; Winograd, S. : Matrix multiplication via arithmetic progressions. In: *Journal of Symbolic Computation* 9 (1990), S. 251–280

[16] Cormen, T. H. ; Leiserson, C. E. ; Rivest, R. L. ; Stein, C. : *Introduction to Algorithms*. B&T, 2001

[17] Cristianini, N. ; Shawe-Taylor, J. : *Support Vector Machines And Other Kernel-based Learning Methods*. Cambridge : Cambridge University Press, 2000

[18] Cucker, F. ; Smale, S. : On the mathematical foundations of learning. In: *American Mathematical Society* 39 (2001), Nr. 1, S. 1–49

[19] D'Agostini, G. : *Bayesian Reasoning in Data Analysis: A Critical Introduction*. World Scientific Publishing, 2003

[20] Dearden, R. ; Friedman, N. ; Andre, D. : Model based Bayesian exploration. In: *Proc. of the Conference on Uncertainty in Artificial Intelligence*, 1999, S. 150–159

[21] Dearden, R. ; Friedman, N. ; Russell, S. J.: Bayesian Q-learning. In: *Proc. of the Innovative Applications of Artificial Intelligence Conference of the Association for the Advancement of Artificial Intelligence*, 1998, S. 761–768

[22] Delage, E. ; Mannor, S. : Percentile optimization in uncertain Markov decision processes with application to efficient exploration. In: *Proc. of the International Conference on Machine Learning*, 2007, S. 225 – 232

[23] Doren, V. J. V.: *Loop Tuning Fundamentals*. Red Business Information, 2003

[24] Drezner, Z. : Computation of the trivariate normal integral. In: *Mathematics of Computation* 63 (1994), S. 289–294

[25] Duff, M. : *Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes*, University of Massachusetts, Diss., 2002

[26] Engel, Y. ; Mannor, S. ; Meir, R. : Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In: *Proc. of the International Conference on Machine Learning*, 2003, S. 154–161

[27] Engel, Y. ; Mannor, S. ; Meir, R. : Reinforcement learning with Gaussian processes. In: *Proc. of the International Conference on Machine learning*, 2005, S. 201–208

[28] Fahlman, S. E.: An empirical study of learning speed in back-propagation networks / Carnegie-Mellon University. 1988. – Forschungsbericht

[29] Gabel, T. ; Riedmiller, M. : Reducing policy degradation in neuro-dynamic programming. In: *Proc. of the European Symposium on Artificial Neural Networks*, 2006, S. 653–658

[30] Garcia, C. E. ; Prett, D. M. ; Morari, M. : Model predictive control: theory and practice. In: *Automatica* (1989), Nr. 25, S. 335–348

[31] Geman, S. ; Bienenstock, E. ; Doursat, R. : Neural networks and the bias/variance dilemma. In: *Neural Computation* 4(1) (1992), S. 1–58

[32] Gerstner, W. ; Kistler, W. M.: *Spiking Neuron Models, Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002

[33] Ghavamzadeh, M. ; Engel, Y. : Bayesian policy gradient algorithms. In: *Advances in Neural Information Processing Systems 19*, 2006, S. 457–464

[34] Ghavamzadeh, M. ; Engel, Y. : Bayesian actor-critic algorithms. In: *Proc. of the International Conference on Machine learning*, 2007, S. 297–304

[35] Giegerich, R. ; Meyer, C. ; Steffen, P. : Towards a discipline of dynamic programming. In: *Informatik bewegt, GI-Edition - Lecture Notes in Informatics* (2002), S. 3–44

[36] Gordon, G. J.: *Approximate Solutions to Markov Decision Processes*, Carnegie Mellon Univsersity, Diss., 1999

[37] Hairer, E. ; Norsett, S. ; Wanner, G. : *Solving Ordinary Differential Equations I.* Springer, 2002

[38] Hans, A. ; Schneegass, D. ; Schäfer, A. M. ; Udluft, S. : Sichere Exploration für Reinforcement-Learning-basierte Regelung. In: *Tagungsband des Workshops für Computational Intelligence der GMA-FA 5.14 Computational Intelligence und der GI-FG Fuzzy-Systeme und Soft-Computing*, 2007, S. 117–131

[39] Hans, A. ; Schneegass, D. ; Schäfer, A. M. ; Udluft, S. : Safe exploration for reinforcement learning. In: Verleysen, M. (Hrsg.): *Proc. of the European Symposium on Artificial Neural Networks*, 2008, S. 413–418

[40] Hastie, T. ; Tibshirani, R. ; Friedman, J. : *The Elements Of Statistical Learning Theory: Data Mining, Inference, and Prediction.* New York : Springer, 2001

[41] Haykin, S. : *Neural Networks : A Comprehensive Foundation.* New York : Macmillan, 1994

[42] Helferich, C. : *Geschichte der Philosophie: Von den Anfängen bis zur Gegenwart und Östliche Denken.* Dtv, 1998

[43] Hinton, G. E. ; McClelland, J. L. ; Rumelhart, D. E.: Distributed representations. In: *Parallel Distributed Processing*, 1986, S. 77–109

[44] Hochreiter, S. : Recurrent neural net learning and vanishing gradient. In: *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems* 6 (1998), Nr. 2, S. 107–116

[45] Hopfield, J. J.: Neural networks and physical systems with emergent collective computational abilities. In: *Proc. of the National Academy of Sciences of the USA* Bd. 79, 1982, S. 2554–2558

[46] Hornik, K. ; Stinchcombe, M. ; White, H. : Multi-layer feedforward networks are universal approximators. In: *Neural Networks* 2 (1989)

[47] Jaakkola, T. ; Jordan, M. I. ; Singh, S. P.: Convergence of stochastic iterative dynamic programming algorithms. In: Cowan, J. D. (Hrsg.) ; Tesauro, G. (Hrsg.) ; Alspector, J. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 6, Morgan Kaufmann Publishers, Inc., 1994, S. 703–710

[48] Jäger, H. : The echo state approach to analysing and training recurrent neural networks. In: *GMD Report 148, GMD - German National Research Institute for Computer Science* (2001)

[49] Jung, T. ; Polani, D. : Kernelizing LSPE(lambda). In: *Proc. of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*, 2007

[50] Kakade, S. : A natural policy gradient. In: *Advances in Neural Information Processing Systems 14*, 2002

[51] Kearns, M. ; Mansour, Y. ; Ng, A. Y.: Approximate planning in large POMDPs via reusable trajectories. In: *Advances in Neural Information Processing Systems 12*, 2000

[52] Keerthi, S. S. ; Shevade, S. K. ; Bhattacharyya, C. ; Murthy, K. R. K.: Improvements to Platt's SMO algorithm for SVM classifier design. In: *Neural Computation* 13 (2001), Nr. 3, S. 637–649

[53] Kohonen, T. : Self-organized formation of topologically correct feature maps. In: *Biological Cybernetics* 43 (1982), S. 59–69

[54] Konda, V. R. ; Tsitsiklis, J. N.: On actor-critic algorithms. In: *SIAM Journal on Control and Optimization* 42(4) (2003), S. 1143–1166

[55] Kap. Maximal margin perceptron In: Kowalczyk, A. : *Advances in Large Margin Classifiers*. MIT Press, 2000, S. 75–113

[56] Kwon, W. H. ; Bruckstein, A. M. ; Kailath, T. : Stabilizing state feedback design via the moving horizon method. In: *International Journal of Control* (1983), Nr. 37, S. 631–643

[57] Lagoudakis, M. G. ; Parr, R. : Least-squares policy iteration. In: *Journal of Machine Learning Research* (2003), S. 1107–1149

[58] Laplace, P.-S. : Mémoire sur la probabilité des causes par les événements. In: *Savants Étranges* 6 (1774), S. 621–656

[59] MacKay, D. J. C.: *Information Theory, Inference, and Learning Algorithms*. Cambridge : Cambridge University Press, 2003

[60] Mannor, S. ; Simester, D. ; Sun, P. ; Tsitsiklis, J. N.: Bias and variance in value function estimation. In: *Proc. of the International Conference on Machine Learning*, 2004

[61] Martinetz, T. : MaxMinOver: A simple incremental learning procedure for support vector classification. In: *Proc. of the International Joint Conference on Neural Networks (IEEE Press)*, 2004, S. 2065–2070

[62] Martinetz, T. ; Labusch, K. ; Schneegass, D. : SoftDoubleMinOver: A simple procedure for maximum margin classification. In: *Proc. of the International Conference on Artificial Neural Networks*, 2005, S. 301–306

[63] Martinetz, T. ; Labusch, K. ; Schneegass, D. : SoftDoubleMaxMinOver: Perceptron-like Trainining of Support Vector Machines. In: *IEEE Transactions on Neural Networks* (akzeptiert, 2008)

[64] Martinetz, T. ; Schulten, K. : A neural-gas network learns topologies. In: *Artificial Neural Networks* (1991), S. 397–402

[65] Moore, A. W. ; Atkeson, C. G.: Prioritized sweeping: Reinforcement learning with less data and less time. In: *Machine Learning* 13 (1993), S. 103–130

[66] Munos, R. : Error bounds for approximate policy iteration. In: *Proc. of the International Conference on Machine Learning*, 2003, S. 560–567

[67] Neapolitan, R. E.: *Learning Bayesian Networks*. Prentice Hall, 2003

[68] Neuneier, R. ; Zimmermann, H. G.: How to train neural networks. In: *Neural Networks: Tricks of the Trade*. Berlin : Springer Verlag, 1998, S. 373–423

[69] Ormoneit, D. ; Sen, S. : Kernel-based reinforcement learning. In: *Machine Learning* 49 (2002), Nr. 2-3, S. 161–178

[70] Pearlmutter, B. : Gradient calculations for dynamic recurrent neural networks: A survey. In: *IEEE Transactions on Neural Networks* 6 (1995), Nr. 5, S. 1212–1228

[71] Peng, J. ; Williams, R. J.: Efficient learning and planning within the DYNA framework. In: *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior, Hawaii*, 1993

[72] Peshkin, L. ; Mukherjee, S. : Bounds on sample size for policy evaluation in Markov environments. In: *Proc. of Annual Conference on Computational Learning Theory, COLT and the European Conference on Computational Learning Theory* Bd. 2111, Springer, Berlin, July 2001, S. 616–629

[73] Peters, J. F. ; Henry, C. : Approximation spaces in off-policy Monte Carlo learning. In: *Engineering Applications of Artificial Intelligence* 20 (2007), Nr. 5

[74] Platt, J. C.: Fast training of support vector machines using sequential minimal optimization. In: *Advances in kernel methods: support vector learning*, MIT Press, Cambridge, MA, USA, 1999, S. 185–208

[75] Popper, K. : *The Logic of Scientific Discovery.* Harper Torch Book, 1968

[76] Poupart, P. ; Vlassis, N. ; Hoey, J. ; Regan, K. : An analytic solution to discrete Bayesian reinforcement learning. In: *Proc. of the International Conference on Machine Learning*, 2006, S. 697 – 704

[77] Prokhorov, D. V. ; Wunsch II, D. C.: Adaptive critic designs. In: *IEEE Transactions on Neural Networks* 8 (1997), Nr. 5, S. 997–1007

[78] Puterman, M. L.: *Markov Decision Processes.* New York : John Wiley & Sons, 1994

[79] Raiffa, H. ; Schlaifer, R. : *Applied Statistical Decision Theory.* Wiley-Interscience, 1961

[80] Rasmussen, C. E. ; Kuss, M. : Gaussian processes in reinforcement learning. In: *Advances in Neural Information Processing Systems 16*, 2003, S. 751–759

[81] Ratliff, P. ; Garbett, P. ; Fischer, W. : The new Siemens gas turbine SGT5-8000H for more costumer benefit. In: *VGB PowerTech* (2007), September

[82] Reitmann, V. : *Reguläre und Chaotische Systeme.* Teubner, Leipzig, 1996

[83] Riedmiller, M. : Neural fitted Q-Iteration - first experiences with a data efficient neural reinforcement learning method. In: *Proceedings of the 16th European Conference on Machine Learning*, 2005, S. 317–328

[84] Riedmiller, M. ; Braun, H. : RPROP - a fast adaptive learning algorithm. In: *Proc. of the International Symposium on Computer and Information Science VII*, 1992

[85] 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

[86] Rumelhart, D. E. ; Hinton, G. E. ; Williams, R. J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), Nr. 9, S. 533–536

[87] Rummery, G. ; Niranjan, M. : On-line Q-learning using connectionist systems / Cambridge University Engineering Department. 1994 (166). – CUED/F-INFENG/TR

[88] Rychetsky, M. : *Algorithms and Architectures for Machine Learning based on Regularized Neural Networks and Support Vector Approaches (Berichte aus der Informatik).* Shaker, 2001

[89] Sahni, S. : Computationally related problems. In: *SIAM Journal on Computing* 3 (1974), S. 262–279

[90] Scales, L. E.: *Introduction to Non-Linear Optimization.* New York : Springer-Verlag, 1985

[91] Schäfer, A. M. ; Zimmermann, H. G.: Recurrent neural networks are universal approximators. In: Kollias, S. (Hrsg.) ; Stafylopatis, A. (Hrsg.) ; Duch, W. (Hrsg.) ; Oja, E. (Hrsg.): *Proc. of the International Conference. Proceedings, Part I*, 2006, S. 632–640

[92] Schäfer, A. M. ; Schneegass, D. ; Sterzing, V. ; Udluft, S. : A neural reinforcement learning approach to gas turbine control. In: *Proc. of the International Joint Conference on Neural Networks*, 2007

[93] Schäfer, A. M. ; Udluft, S. ; Zimmermann, H.-G. : A recurrent control neural network for data-efficient reinforcement learning. In: *Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, S. 151–157

[94] Schneegass, D. : On the bias of batch Bellman residual minimisation. In: *Neurocomputing* (akzeptiert, 2008)

[95] Schneegass, D. ; Labusch, K. ; Martinetz, T. : MaxMinOver regression: A simple incremental approach for support vector function approximation. In: Kollias, S. (Hrsg.) ; Stafylopatis, A. (Hrsg.) ; Duch, W. (Hrsg.) ; Oja, E. (Hrsg.): *Proc. of the International Conference. Proceedings, Part I*, 2006, S. 150–158

[96] Schneegass, D. ; Martinetz, T. ; Clausohm, M. : OnlineDoubleMaxMinOver: A simple approximate time and information efficient online support vector classification method. In: Verleysen, M. (Hrsg.): *Proc. of the European Symposium on Artificial Neural Networks*, 2006, S. 575–580

[97] Schneegass, D. ; Schäfer, A. M. ; Martinetz, T. : The intrinsic recurrent support vector machine. In: Verleysen, M. (Hrsg.): *Proc. of the European Symposium on Artificial Neural Networks*, 2007, S. 325 – 330

[98] Schneegass, D. ; Udluft, S. ; Martinetz, T. : Kernel rewards regression: An information efficient batch policy iteration approach. In: *Proc. of the IASTED Conference on Artificial Intelligence and Applications*, 2006, S. 428–433

[99] Schneegass, D. ; Udluft, S. ; Martinetz, T. : Explicit kernel rewards tegression for data-efficient near-optimal policy identification. In: Verleysen, M. (Hrsg.): *Proc. of the European Symposium on Artificial Neural Networks*, 2007, S. 337 – 342

[100] Schneegass, D. ; Udluft, S. ; Martinetz, T. : Improving optimality of neural rewards regression for data-efficient batch near-optimal policy identification. In: *Proc. of the International Conference on Artificial Neural Networks*, 2007, S. 109–118

[101] Schneegass, D. ; Udluft, S. ; Martinetz, T. : Neural rewards regression for near-optimal policy identification in Markovian and partial observable environments. In: Verleysen, M. (Hrsg.): *Proc. of the European Symposium on Artificial Neural Networks*, 2007, S. 301 – 306

[102] Schneegass, D. ; Udluft, S. ; Martinetz, T. : Uncertainty propagation for quality assurance in reinforcement learning. In: *Proc. of the International Joint Conference on Neural Networks*, 2008, S. 2589–2596

[103] Schrauwen, B. ; Verstraeten, D. ; Campenhout, J. V.: An overview of reservoir computing: Theory, applications and implementation. In: Verleysen, M. (Hrsg.): *Proc. of the European Symposium on Artificial Neural Networks*, 2007, S. 471–482

[104] Smallwood, R. D. ; Sondik, E. J.: The optimal control of partially observable Markov processes over a finite horizon. In: *Operations Research* 21 (1973), Nr. 5, S. 1071–1088

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

[106] Sutton, R. S.: First results with DYNA, an integrated architecture for learning, planning, and reacting. In: *Neural Networks for Control* (1990)

[107] Sutton, R. S. ; Barto, A. G.: *Reinforcement Learning: An Introduction*. Cambridge : MIT Press, 1998

[108] Sutton, R. S. ; McAllester, D. ; Singh, S. ; Mansour, Y. : Policy gradient methods for reinforcement learning with function approximation. In: *Advances in Neural Information Processing Systems 12*, 2000

[109] Tesauro., G. J.: TD-gammon, a self-teaching Backgammon program, achieves master-level play. In: *Neural Computation* 6(2) (1994), S. 215–219

[110] Timmer, S. ; Riedmiller, M. : Fitted Q iteration with CMACs. In: *Proc. of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*, 2007, S. 1–8

[111] Tresp, V. : The wet game of chicken / Siemens AG, CT IC 4. 1994. – Forschungsbericht

[112] Tsitsiklis, J. N. ; Roy, B. V.: An analysis of temporal difference learning with function approximation. In: *IEEE Transactions on Automatic Control* 42(5) (1997), S. 674–690

[113] Turing, A. M.: Computing machinery and intelligence. In: *Mind* (1950), S. 433–460

[114] van der Vaart, A. W.: *Asymptotic Statistics*. Cambridge University Press, 1998

[115] Valiant, L. G.: A theory of the learnable. In: *Communications of the ACM 27* (1984), S. 1134–1142

[116] Vapnik, V. N.: *Statistical Learning Theory*. New York : John Wiley & Sons, Inc., 1998

[117] Vapnik, V. N.: *The Nature of Statistical Learning Theory*. Cambridge : Cambridge University Press, 2000

[118] Verschiedene Autoren: *Guide to the Expression of Uncertainty in Measurement*. International Organization for Standardization, 1993

[119] Watkins, C. : *Learning from Delayed Rewards*, Cambridge University, Diss., 1989

[120] Watkins, C. ; Dayan, P. : Q-Learning. In: *Machine Learning* 8 (1992), Nr. 3

[121] Werbos, P. J.: *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Harvard University, Diss., 1974

[122] Xu, X. ; Hu, D. ; Lu, X. : Kernel-based least squares policy iteration for reinforcement learning. In: *IEEE Transactions on Neural Networks* 18 (2007), Nr. 4, S. 973–992

[123] Zeugmann, T. : Data mining / University of Lübeck. 2003. – Lecture Notes

[124] Kap. Identification and forecasting of large dynamical systems by dynamical consistent neural networks In: Zimmermann, H. G. ; Grothmann, R. ; Schäfer, A. M. ; Tietz, C. : *New Directions in Statistical Signal Processing: From Systems to Brain*. MIT Press, 2006, S. 203–242

[125] Kap. Neural network architectures for the modeling of dynamical systems In: Zimmermann, H. G. ; Neuneier, R. : *A Field Guide to Dynamical Recurrent Networks*. 2001, S. 311–350

[126] Kap. Modeling of dynamical systems by error correction neural networks In: Zimmermann, H. G. ; Neuneier, R. ; Grothmann, R. : *Modeling and Forecasting Financial Data, Techniques of Nonlinear Dynamics*. Kluwer Academic Publishers, 2002, S. 237–263

Index

Curriculum vitae

Name:		Daniel Schneegaß
Geburtsdatum, -ort:		14. Juli 1980 in Rathenow
Nationalität:		deutsch
Schulausbildung:	09/87 - 08/91 09/91 - 07/99	**Grundschule Neubrandenburg** **Sportgymnasium Neubrandenburg**
Schulabschluss:	07/99	Abitur, Note 1.3
Grundwehrdienst:	11/99 - 08/00	**Stabskompanie, Panzeraufklärungs-bataillon Eutin**
Studium:	10/00 - 05/05	**Universität zu Lübeck** Informatik (Bioinformatik/Biomathematik)
	12/04 - 05/05	Diplomarbeit: Der DoubleMaxMinOver Approach zur Bestimmung der Support Vektoren bei Klassifikation und Regression mit Anwendungen in der Prozessindustrie
	05/05	Abschluss Note 1.3
Promotion:	seit 06/05	**Siemens AG, Corporate Technology**, München Promotion in der Zentralen Forschung, Information & Communications, Abteilung Learning Systems in Kooperation mit dem Institut für Neuro- und Bioinformatik, Universität zu Lübeck, Thema: Steigerung der Informationseffizienz im Reinforcement Learning
Berufserfahrung:	07/01 - 05/05	**Clausohm Software GmbH**, Neverin Softwareentwicklung (ca. 25 h/mt.) im Bereich Automatisierungstechnik und Automobilindustrie

	01/01 - 07/04	**Universität zu Lübeck** Verschiedene Tätigkeiten als Hilfswissenschaftler und Freier Mitarbeiter
	seit 06/05	**Siemens AG, Corporate Technology**, München Wissenschaftliche Mitarbeit in der Zentralen Forschung, Information & Communications, Abteilung Learning Systems Mitarbeit an internen Forschungsprojekten im Bereich Reinforcement Learning und an verschiedenen Bereichsprojekten
Praktika:	07/99 - 08/01	**Sportgymnasium Neubrandenburg** Entwicklung eines Stundenplanprogrammes (Time Table Manager)
	07/02 - 08/02	**GSF Forschungszentrum für Umwelt und Gesundheit**, München Mitarbeit als Werkstudent am Projekt Differenzielle Optische Absorptionsspektroskopie am Institut für Biomathematik und Biometrie
	04/02 - 11/02 09/03 - 11/03	**Universität zu Lübeck** Vorbereitung auf und Teilnahme am Northwestern European International Collegiate Programming Contest (ICPC) der Association of Computing Machinery (ACM) in Delft, Niederlande und Lund, Schweden
Auszeichnungen:	02/99	Anerkennung der 38. Mathematik-Olympiade des Bundeslandes Mecklenburg-Vorpommern
	11/03	Honorable Mention (mit Jens Keiner und Hanno Schumacher) der Northwestern European ICPC der ACM, 2003, Lund, Schweden
Stipendien:	09/06	Reisestipendium der European Neural Network Society zur Teilnahme an der International Conference on Artifical Neural Networks in Athen, 2006