Karlsruher Institut für Technologie

Maschinelles Lernen II - Fortgeschrittene Verfahren
V06 Reinforcement Learning II - Erweiterte Methoden

Sommersemester 2017

Prof. Dr. J.M. Zöllner, Prof. Dr. R. Dillmann, M.Sc. Peter Wolf

INSTITUT FÜR ANGEWANDTE INFORMATIK UND FORMALE BESCHREIBUNGSVERFAHREN
INSTITUT FÜR ANTHROPOMATIK UND ROBOTIK

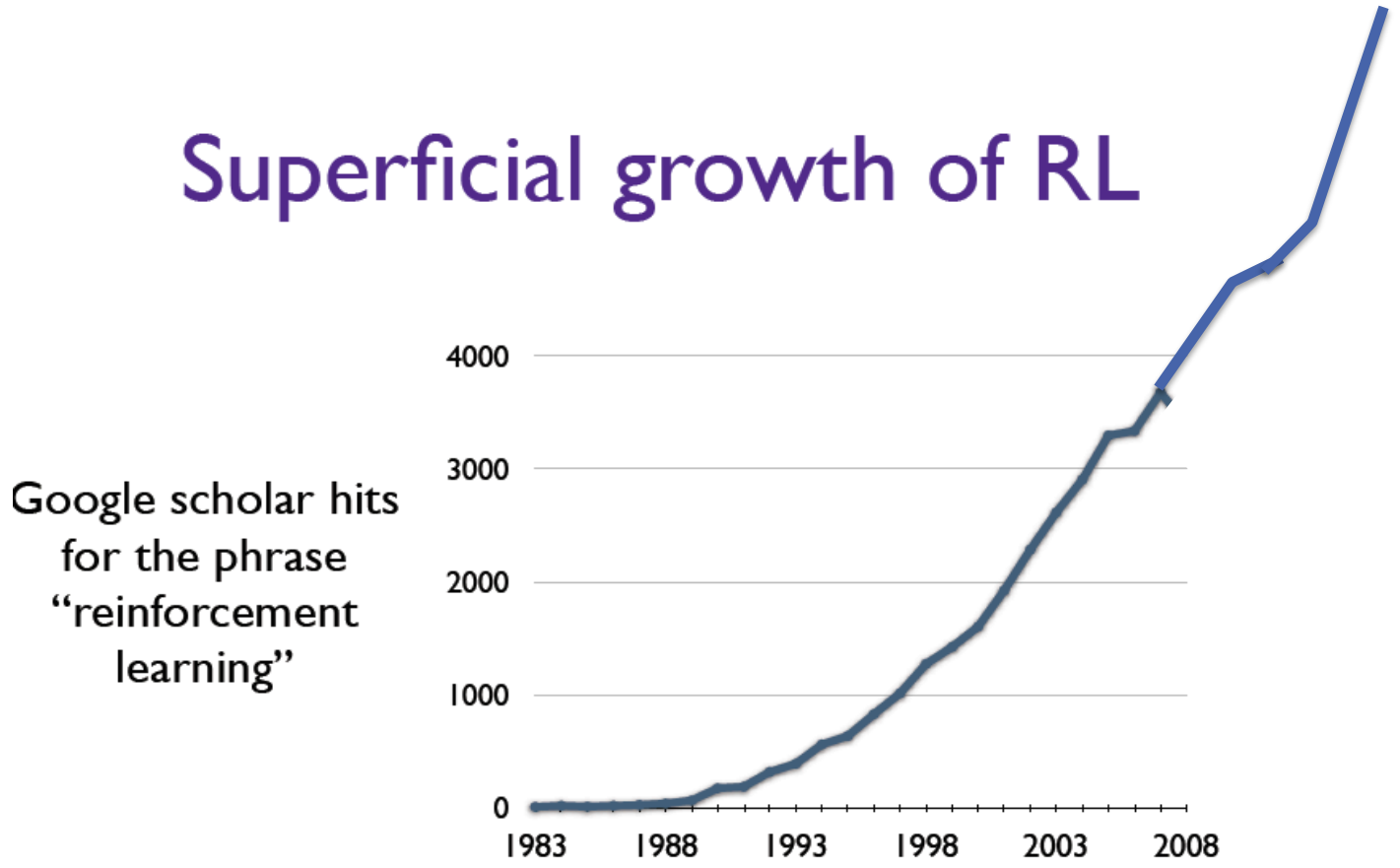
Inhalt

- Wdh.: Was ist RL?
- Problematik RL

- Erweiterte Methoden
  - Effiziente Funktionsapproximation
  - Hierarchisches Reinforcement Learning
    - Options

- Deep Reinforcement Learning
  - Deep Q-Netzwerke

RL – „Statistik“

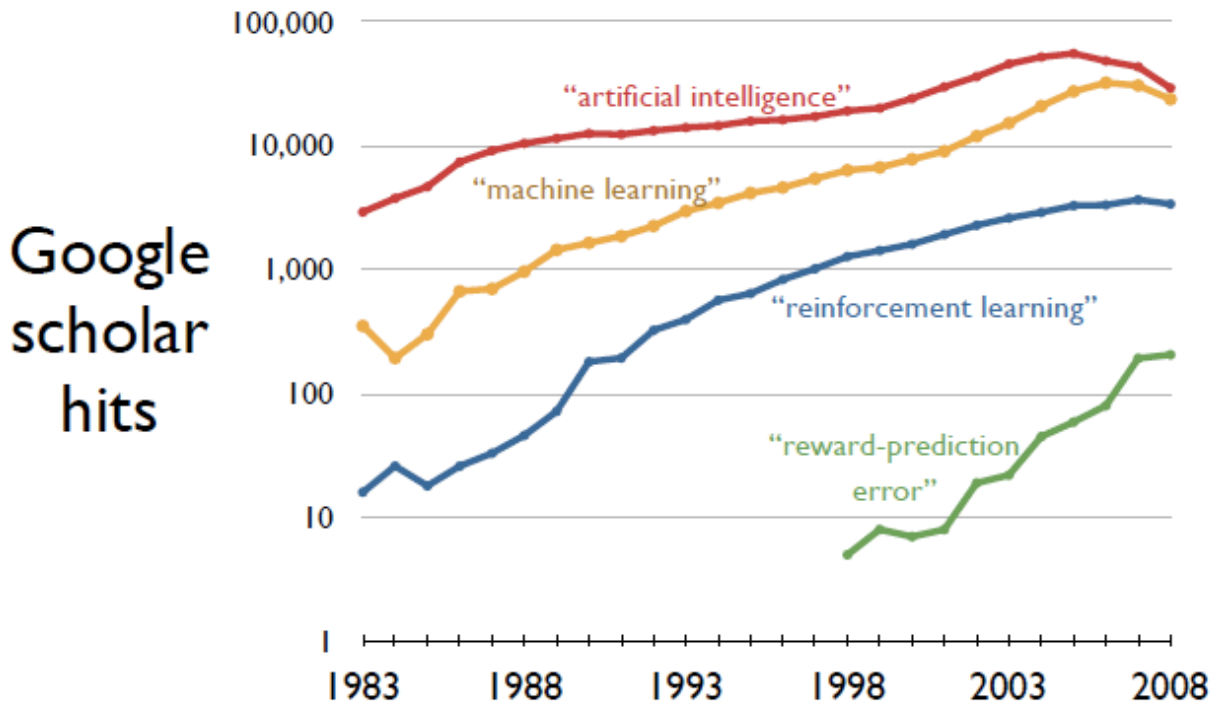


2012: 777000
2013: 900.000
2014: 1.080.000

Sutton 08

RL – „Statistik“

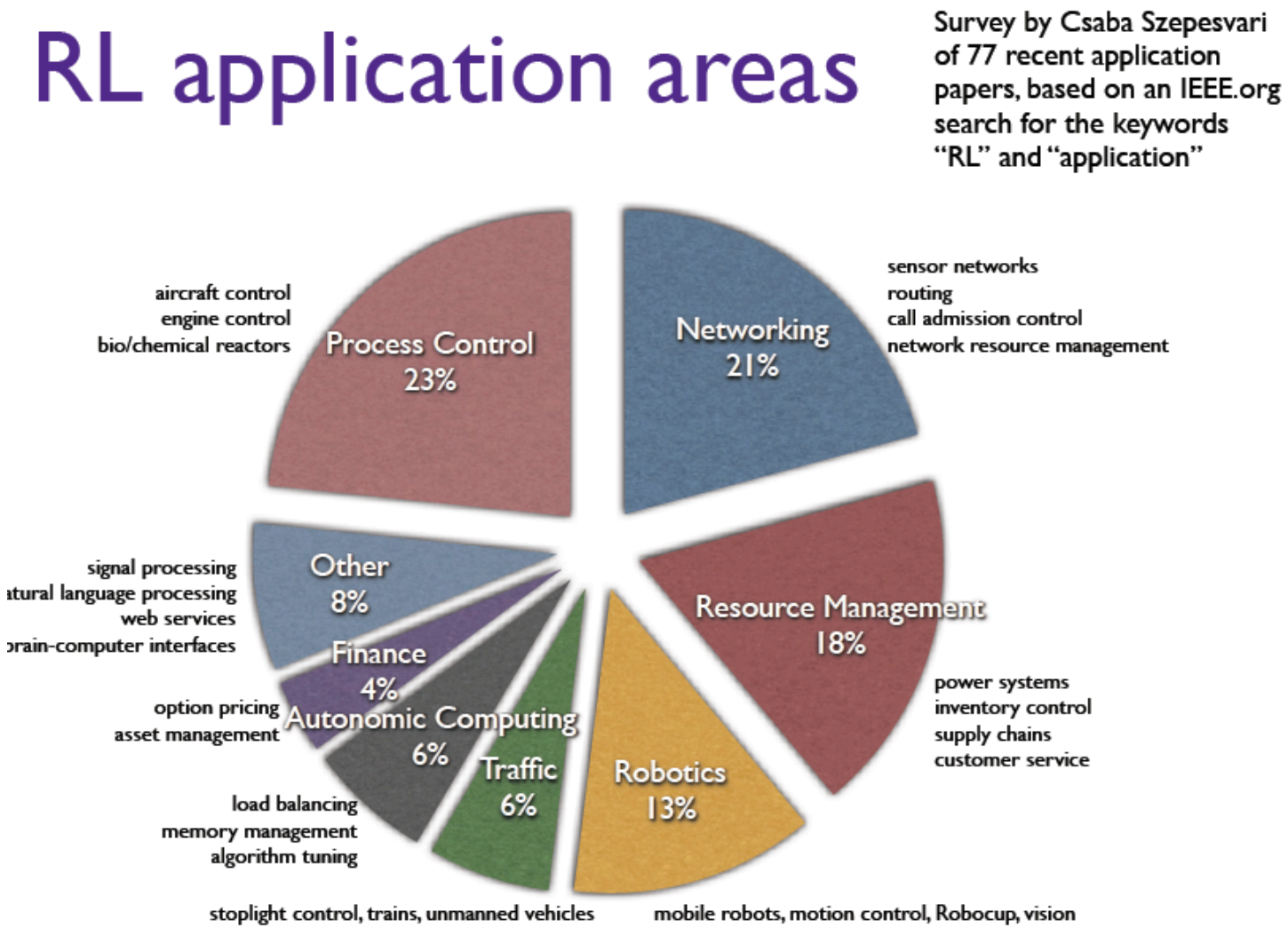
Vs. other buzz-words



ML 2013 - 3.010.000
RL 2013 - 900.000
SVM 2013 - 2.040.000
AI 2013 - 1.920.000

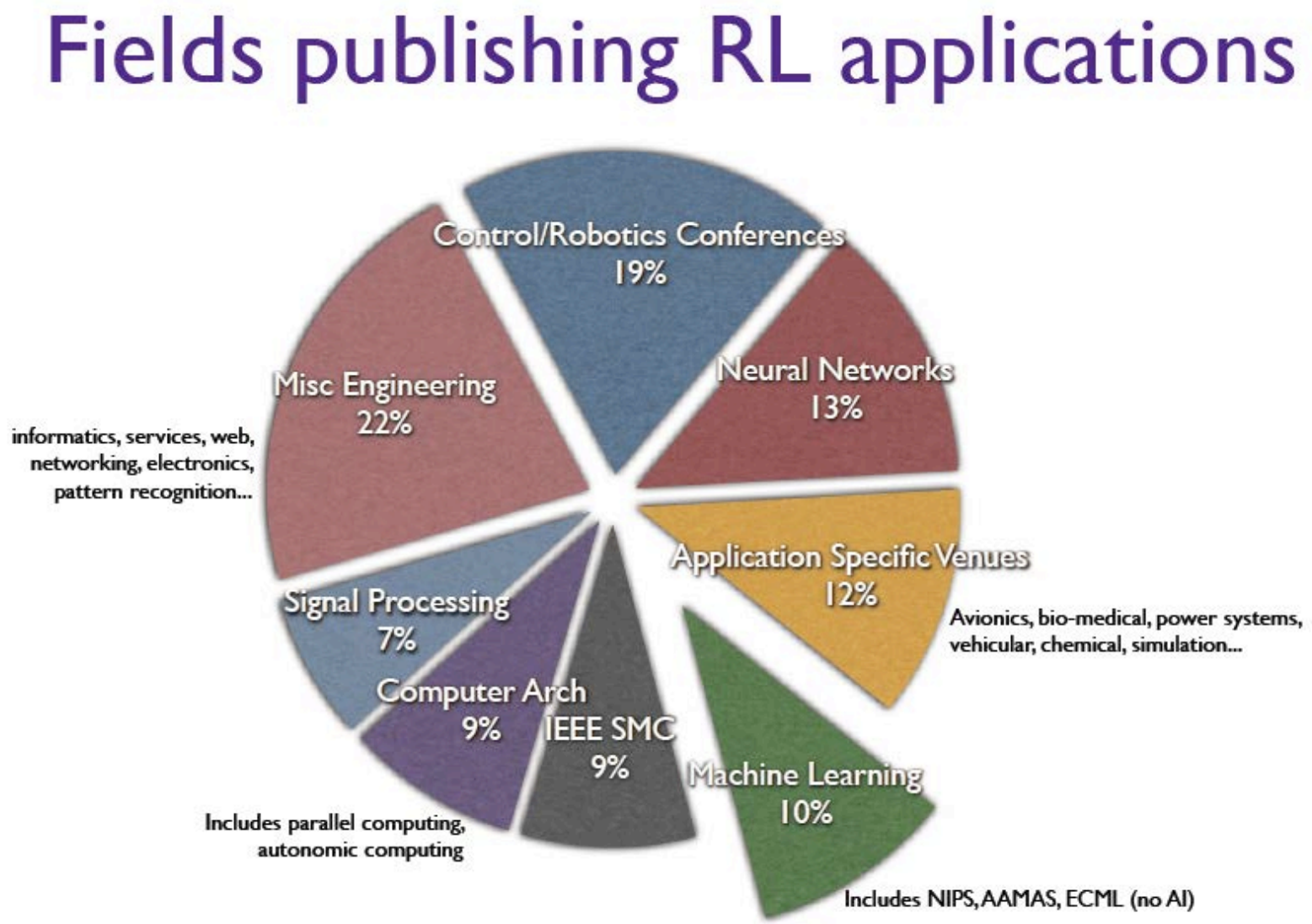
Sutton 08

RL – Anwendungsfelder



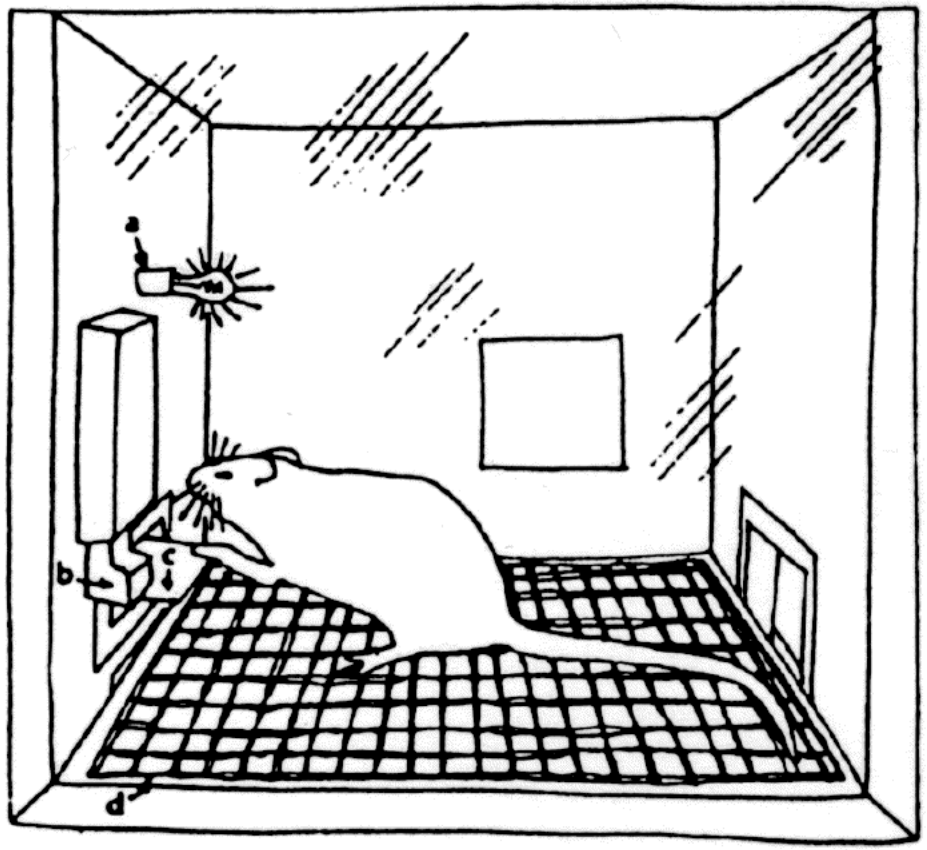
Sutton 08

RL – relevante Themen



Sutton 08

Wieso RL? - Lernen mit Belohnung



a) Licht
b) Futtermagazin
c) Hebel
d) elektr. Rost

Abb. 45. Skinner Box
(aus: Lefrançois, 1976, S. 63)

Markov decision process (deterministisch)

- „Autonomer Agent & Umwelt“:
  - Zustandsgetriebener Prozess
  - „Sensorik“ – Erfassung von Zuständen $s_t \in S$
  - „Aktorik“ – Einwirkung auf die Umwelt durch Aktionen $a_t \in A$

  - Zustandsänderungen $\delta : (S \times A) \rightarrow S$

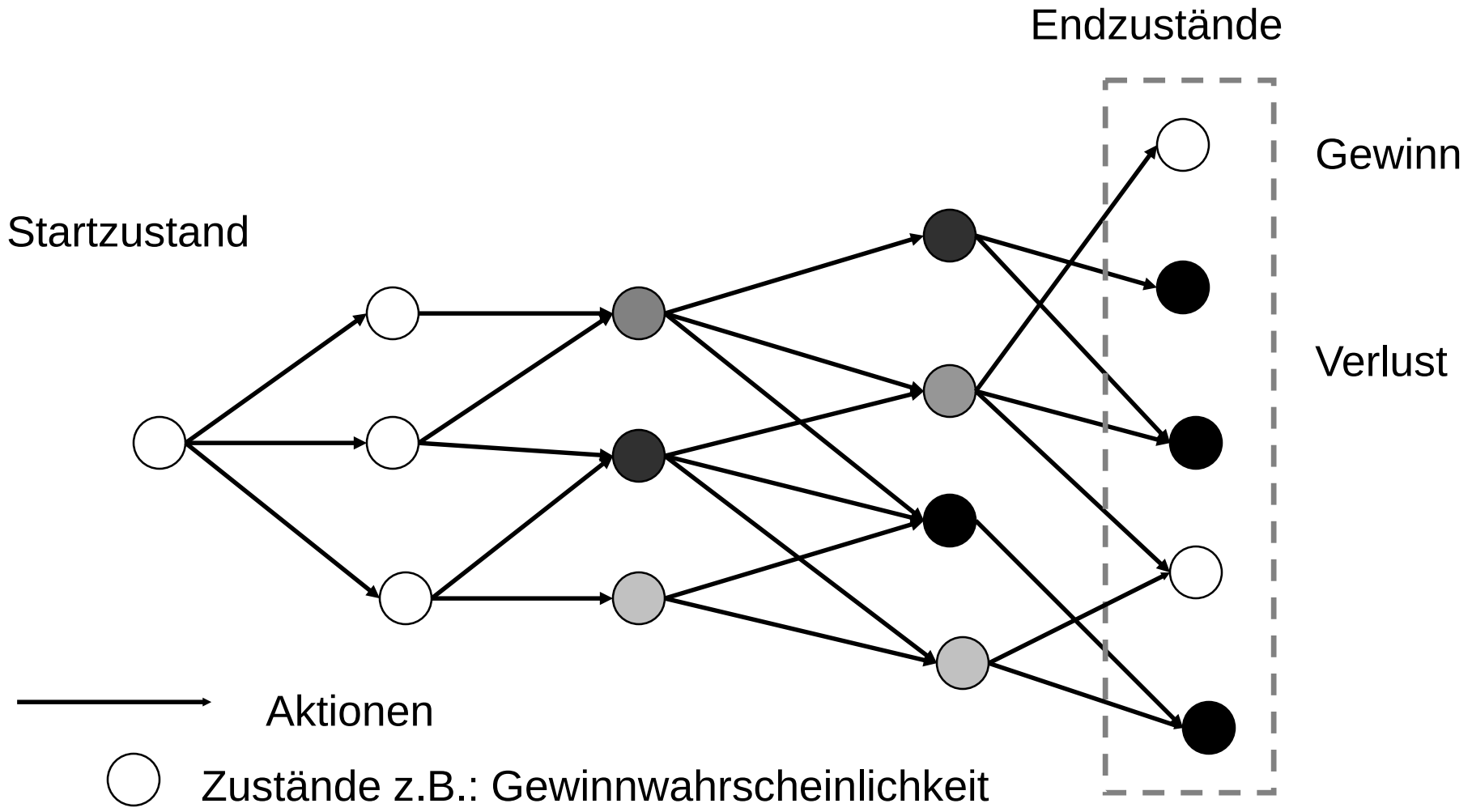
    $\delta(s_t, a_t) = s_{t+1}$

  - Markov-Bedingung: keine Abhängigkeit von der Vergangenheit

- Bewertung von Aktionen $r : (S \times A) \rightarrow R$

  $r(s_t, a_t) = r_t$

Markov decision process



Episodischer MDP

= MDP mit „absorbierenden“ Endzuständen

- Charakterisiert durch: (S, A, P, R, G, s_o)
  - S : Menge von Zuständen.
  - A : Menge möglicher Aktionen.
  - P : Probabilistisches Transitionmodel. $P(s'|s,a)$*
  - R : Reward Modell $R(s)$*
  - G : Endzustände (goal states)
  - s_o : Startzustand
  - γ : Discount Faktor ← endlicher Horizont

* Markov Bedingung:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(s_{t+1}|s_t, a_t)$$

$$P(r_t|s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(r_t|s_t, a_t)$$

Strategielernen - Policy learning

Gesucht: $s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \dots \xrightarrow[r_{n-1}]{a_{n-1}} s_n$

←→ finde die (optimale) Zielfunktion (target function)

$$\pi : S \to A, \quad \pi(s_t) = a_t$$

so dass die akkumulierte Bewertung (zum Ziel hin)

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

maximiert wird

Gewichtung der Bewertungen (Diskontierungsfaktor) $0 \leq \gamma < 1$

0: aktuelle Aktionsbewertung ist wichtig (1-step)

> 0: zukünftige (letzte) Bewertungen werden berücksichtigt (n-step)

Die V- und Q-Funktion (value- bzw state-action function) – Bellmann Gleichungen

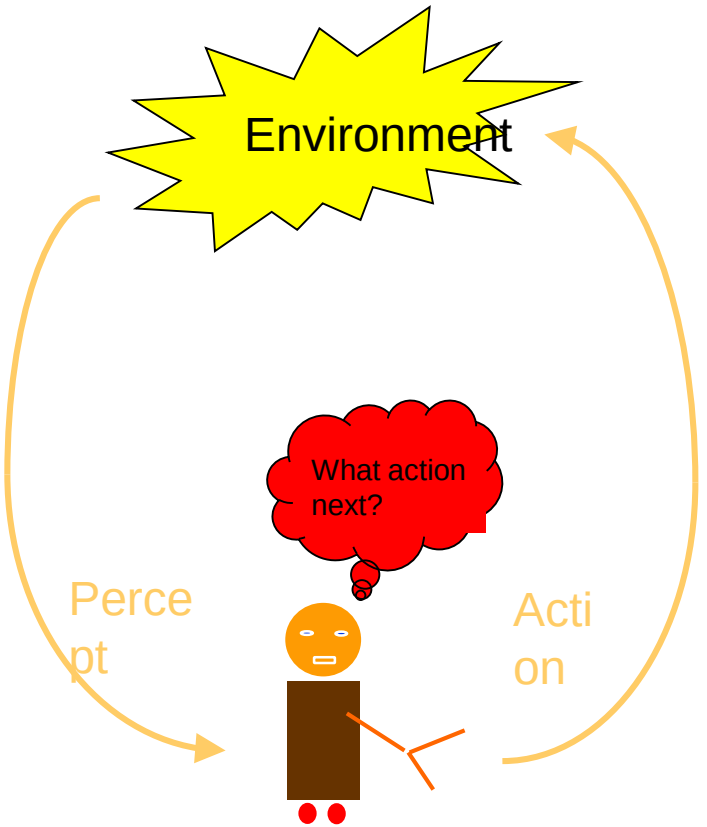
$$V(s) = \max_a \left[r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right]$$

Q(s,a) maximale Bewertung,
die erreicht werden kann
im Zustand s durch die Aktion a

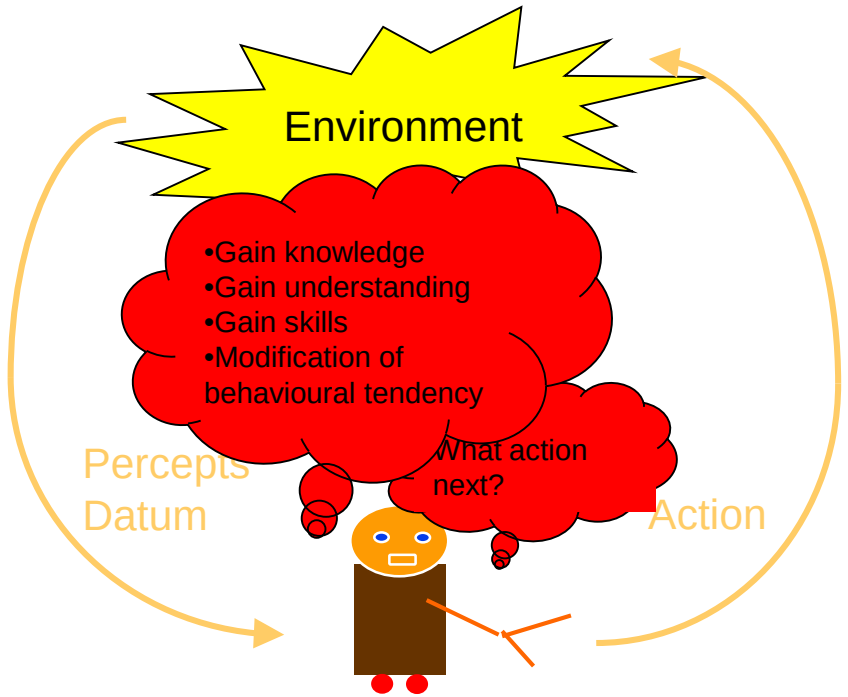
$$Q(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$$

$$V^*(s) = \max_a (Q^*(s,a))$$

rekursiv: $$Q(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$



RL: Entscheiden während des Lernens



Problem oft:
P – unbekannt (Value Iteration nicht möglich)

r – unbekannt (z.B. Eligibility verwenden)

Idee: Lerne $\hat{V}^*(s)\ \forall s$ oder $\hat{Q}(s,a), \quad \forall (s,a) \in S \times A$

Wähle beste Aktion anhand einer Strategie z.B.:

$$\pi^*(s) = \arg\max_a Q(s,a)$$

RL: Entscheiden während des Lernens

Q-Lernen Algorithmus

Ziel: finde Schätzung $\hat{Q}(s,a)$ der absoluten Funktion $Q(s,a)$

Lernen:

In

W

do

Problem

- Erweiterungen nötig (Lernrate)

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha[r + \gamma \max \hat{Q}(s',a') - \hat{Q}(s,a)]$$

- Langsame Konvergenz bei Q-Lernen ➔ SARSA
- SARSA – Verwendung der Policy ➔ Zielwerte ändern sich ständig während des Lernvorganges

➔Funktionsapproximation wird instabil und nur unter gewissen Randbedingungen (z.B. abnehmende Lernrate) konvergent

Idee:

18 Institut für Anthropomatik (IFA) Prof. Dr.-Ing. R. Dillmann *Maschinelles Lernen* Reinforcement Learning Fakultät für Informatik

a

Erweiterte Ansätze des RL

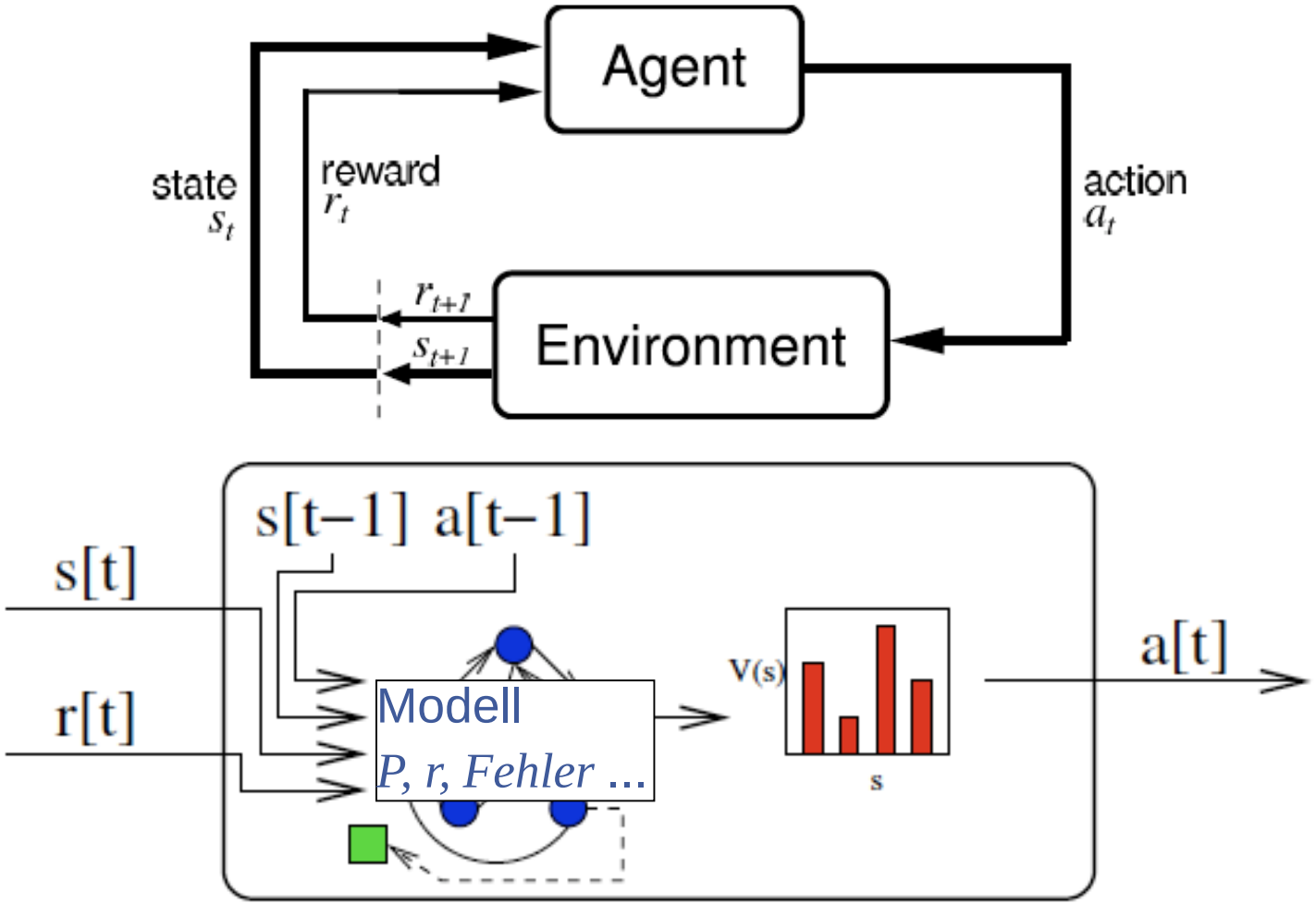
- Wozu noch selber denken? - Wichtigste Gründe ...
  - Verfahren benötigen z.T. Prozesswissen
  - Konvergenz nur für endliche Zustandsräume!
  - Große Zustandsräume schwer zu explorieren!
  - Große Zustandsräume erfordern gute Generalisierung!

- Trends im RL
  - Effiziente Funktionsapproximation
  - Hierarchisches RL
  - Deep RL

Was tun? - Grundsätzliche Ansätze

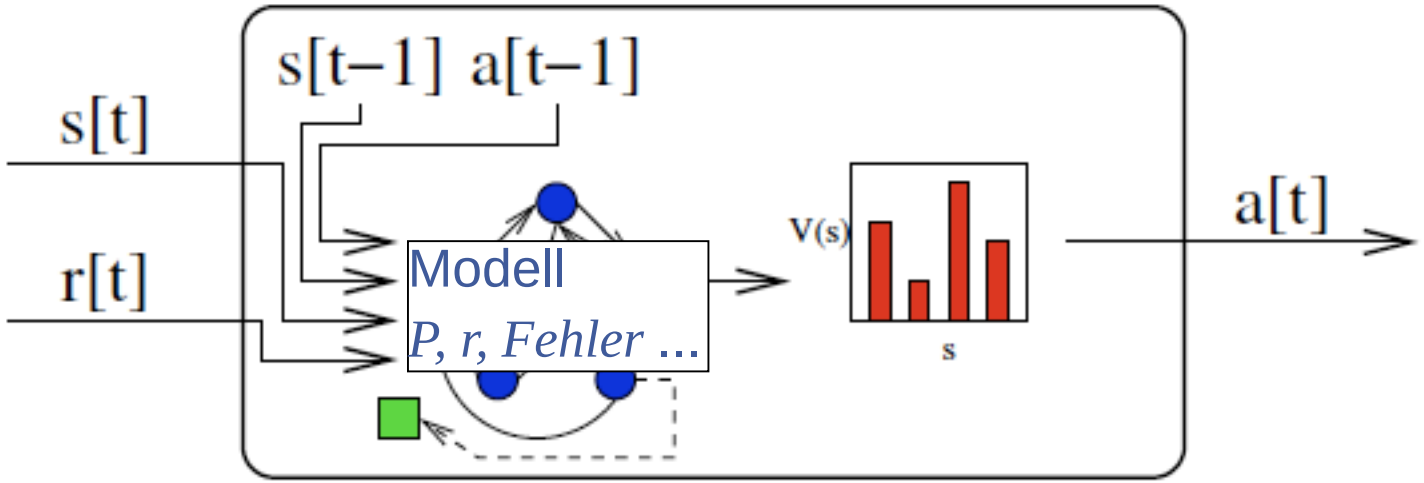
- Grundlagen
  - Bellmann Gleichungen
- Modellbasierter Lernansatz:
  - Grundannahme über ein Modell von P, r
  - **Lerne Modelle (Transitionsmodell, Reward-Modell,....)**
  - Wende MDP an
  - Methoden
    - E^3, R-Max, Value Iteration, Policy Iteration
- Modellfreier Lernansatz:
  - Effizientes, implizites Lernen von V,Q,P,r,....
  - Methoden
    - Q-Learning, SARSA, TD, Monte Carlo Ansatz
    - Kernel based RL

Grundsätzliche Ansätze: Modellbasiert



Grundsätzliche Ansätze: Modellbasiert Lernen

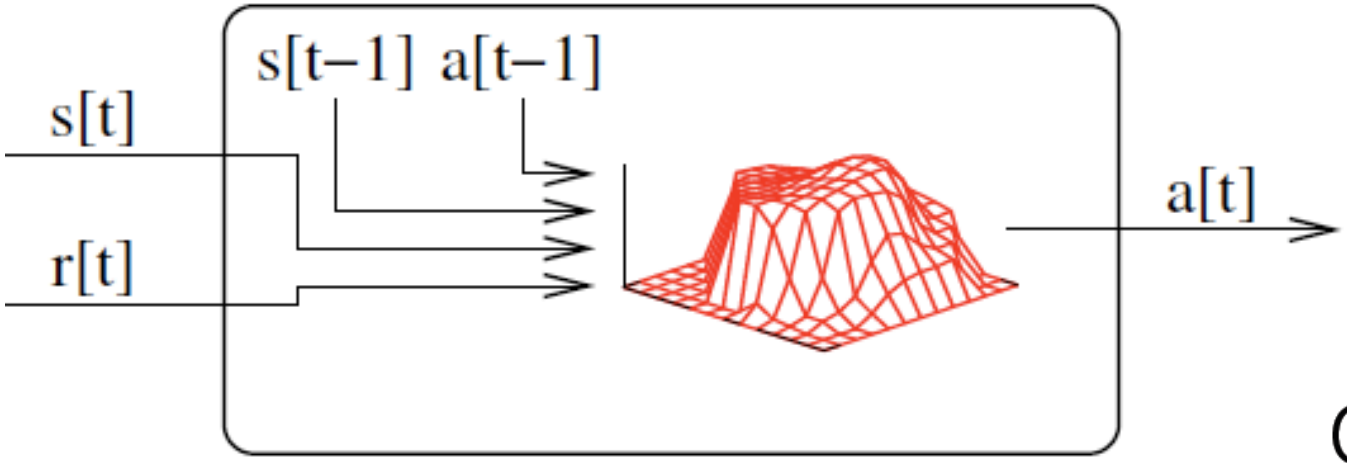
- Modell des Prozesses wird zunächst gelernt
  - P : Probabilistisches Transitionmodel. $P(s(t+1)|s(t),a(t))$
  - R : Reward Modell $R(s)$
  - S : Menge von (diskreten) Zustände
  - A : Menge möglicher (diskreter) Aktionen
- Anhand der Modelle wird z.B. die Exploration des RL durchgeführt um z.B. $V(s)$ zu lernen

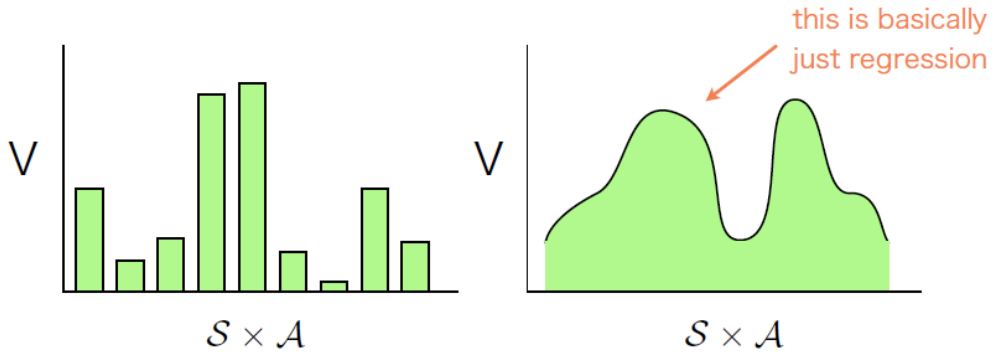


Grundsätzliche Ansätze

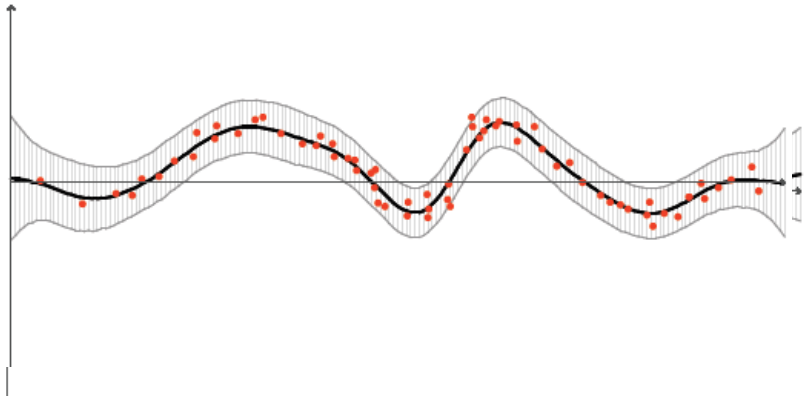
- Grundlagen
  - Bellman Gleichung
- Modellbasierter Lernansatz:
  - Grundannahme über ein Modell von P, r
  - Lerne Modelle (Transitionsmodell, Reward-Modell,....)
  - Wende MDP an
  - Methoden
    - E^3, R-Max, Value Iteration, Policy Iteration
- Modellfreier Lernansatz:
  - Implizites Lernen von V,Q,P,r,....
  - Methoden
    - Q-Learning, SARSA, TD, Monte Carlo Ansatz
    - Kernel based RL

Grundsätzliche Ansätze: Modellfrei → Funktionsapproximation





Gaussian Processes



- Aus wenigen Stützpunkten Funktion effizient lernen

[Reisinger 08] – Online Kernel Selection

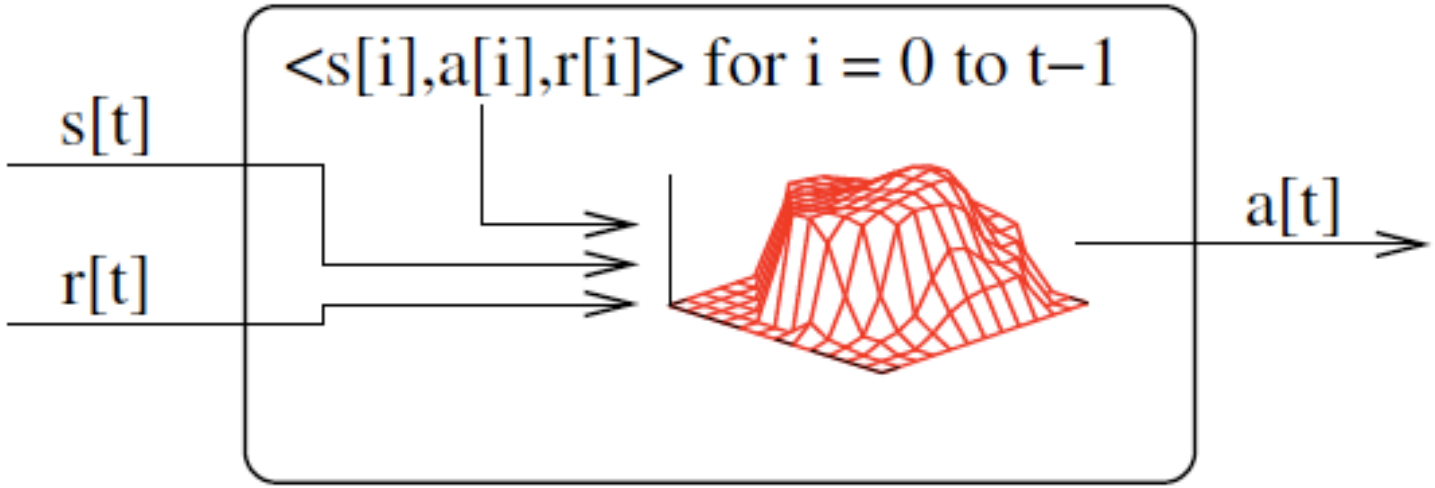
Effiziente Funktionsapproximation

- Große Zustandsräume erfordern eine gute Generalisierung der gelernten V / Q-Funktion
  - i.A inkrementelles Lernen nötig
- Grundlegende Idee (Q – Lernen) : Überwachtes Lernen

$$\{((s_1, a_1), Q(s_1, a_1)), ((s_2, a_2), Q(s_2, a_2)), ..., ((s_M, a_M), Q(s_M, a_M))\}$$

- Endliche Trainingsmenge aus Eingaben und Zielwerten wird durch einen Funktionsapproximator auf eine Funktion abgebildet

$$\hat{Q} : (S \times A) \rightarrow \mathbb{R}$$



Realisierung: Fitted Q-Iteration

- Pseudo Code
  - Require: Q-Funktion $\hat{Q} : (S \times A) \rightarrow \mathbb{R}$
  - loop
    - Berechne Strategie π aus $\hat{Q}$ (z.B. ϵ-greedy)
    - Sampling von Übergängen (s_t, a_t, r_t, s_{t+1}) mit $a_t = \pi(s_t)$
    - Erstellung der Trainingsmenge $\{(s_t, a_t), r_t + \alpha \max_{a \in A} \hat{Q}(s_{t+1}, a)\}$
    - Trainieren eines Funktionsapproximators auf Trainingsmenge ergibt eine neue (aktuelle) Approximation $\hat{Q}$
  - end loop
- Vorteile
  - Für bestimmte Klassen von Funktionsapproximatoren lässt sich Konvergenz beweisen (z.B. im Zusammenhang mit Linearisierung/ Diskretisierung)
  - Generell stabilere Approximation der Q-Funktion und besonders dateneffiziente Exploration (d.h. die Fähigkeit anhand einer sehr begrenzten Zahl von Interaktionen zu lernen)

Inhalt

- Wdh. Was ist RL?
- Problematik RL

- Erweiterte Methoden
  - Effiziente Funktionsapproximation
  - Hierarchisches Reinforcement Learning
    - Options

- Deep Reinforcement Learning
  - Deep Q-Netzwerke

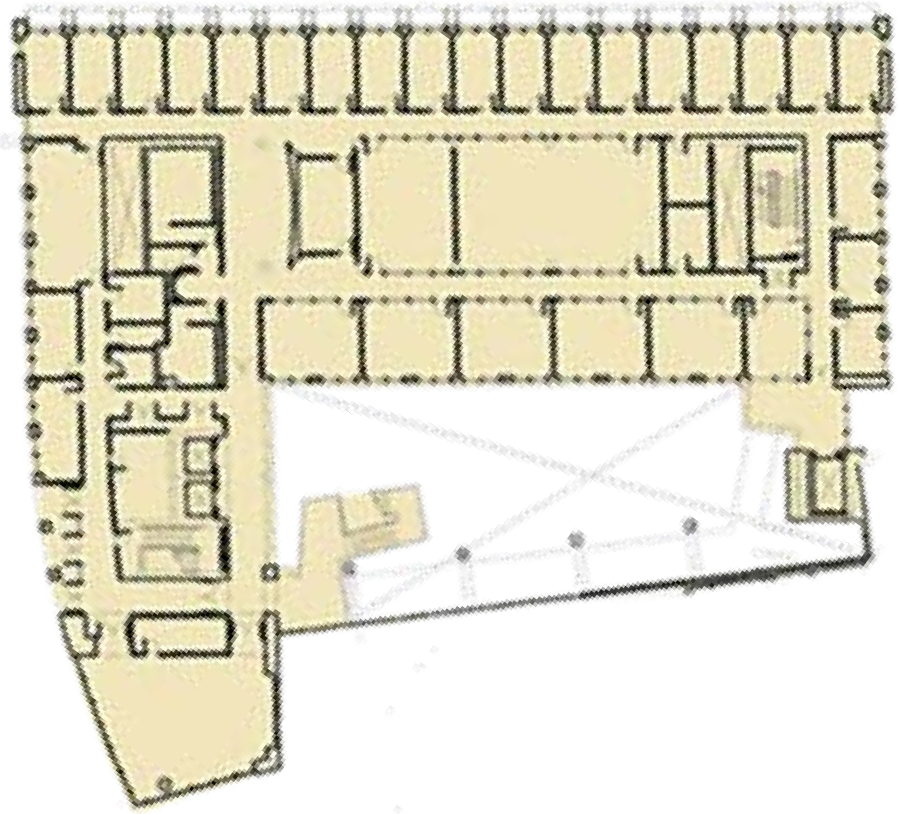
Hierarchisches RL

- Wieso hierarchisch?

- Einfaches Beispiel: Printerbot
  - Roboter der die Aufgabe hat: Vom Drucker - Ausdruck holen
  - States (S) : $\{\text{loc}, \text{has-robot-printout}, \text{user-loc}, \text{has-user-printout}\}, \text{map}$
  - Actions (A) : $\{\text{move}_n, \text{move}_s, \text{move}_e, \text{move}_w, \text{extend-arm}, \text{grab-page}, \text{release-pages}\}$
  - Reward (R) : if h-u-po +20 else -1
  - Goal (G) : All states with h-u-po true.
  - Start state : A state with h-u-po false.

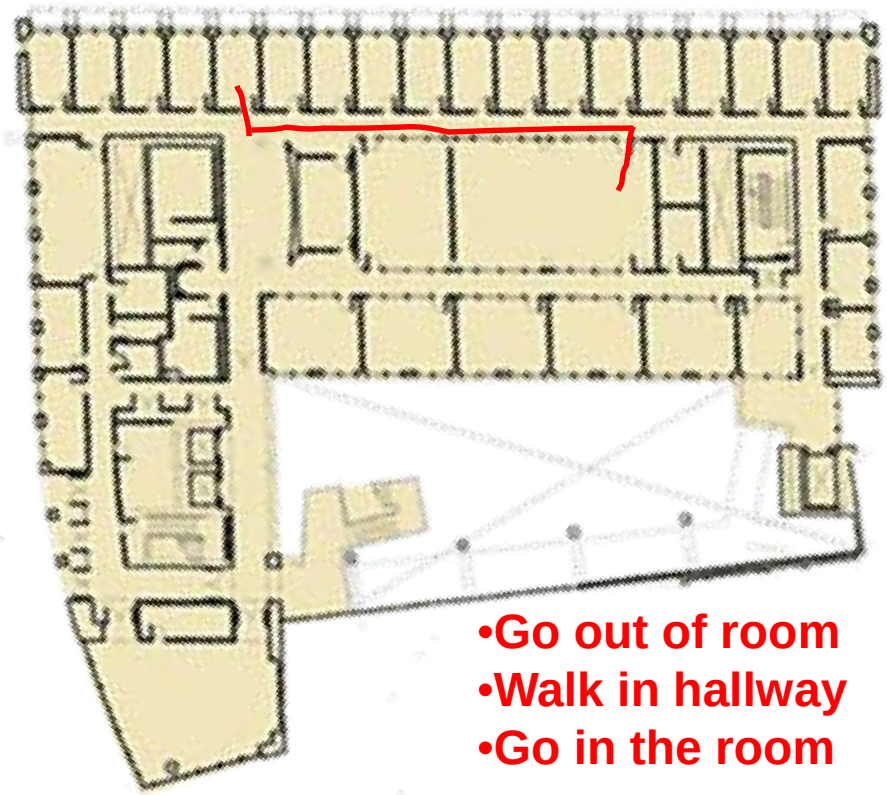
Printerbot - Klassischer Ansatz Problematik

- Eine typische Umgebung (Uni. Washington)
  - Paul G. Allen Center hat 85000 ft^2 ~7600 m^2
  - Pro Etage ~ 85000/7 ~ ~12000 ft^2 ~1000 m^2
  - Diskretisierung pro Etage: 12000 Teilräume
  - Zustandsraum (ohne Karte, z.B. Belegtheit): 12000*2*12000*2 → sehr groß !!!!
- Wie kann man dabei eine sinnvolle Strategie lernen?
  - Wie macht es der Mensch?



Printerbot – Typische Aktionen

- Mensch verwendet „(Teil-)Pläne“, die Modularität auf unterschiedlichen Ebenen beinhalten:
  - Einen Raum verlassen ist identischen mit einen anderen Raum verlassen
  - Navigation ist nicht abhängig davon ob der Roboter einen Ausdruck hat oder nicht
  - ....

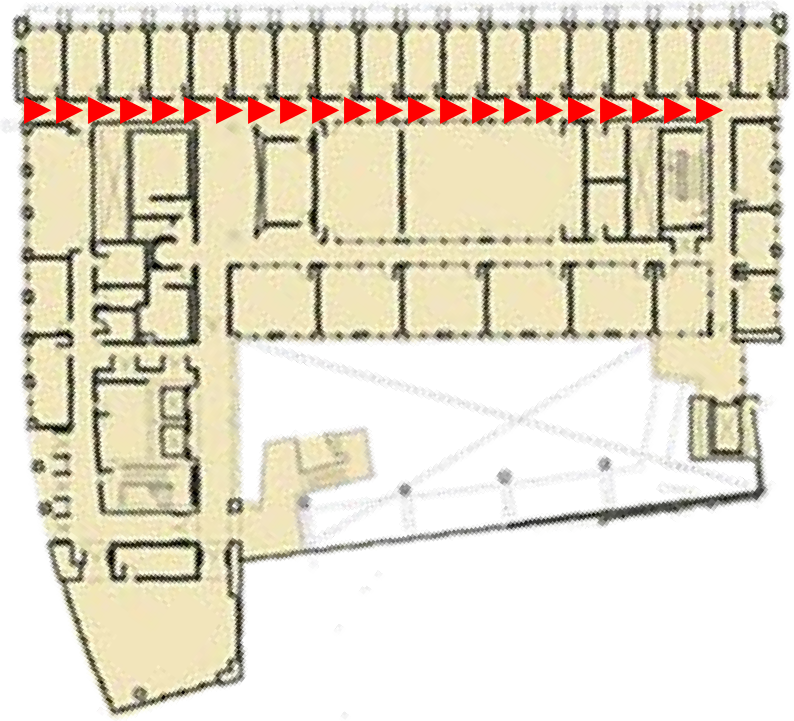


Hierarchisches RL - Zielsetzung

- Möglichkeit um spezielle Strukturen (Hierarchien) in den MDP einzufügen
  - Verhalten, Skills, Unteraufgaben
  - Z.B.: Ausdruck aufnehmen, Kollisionsfreie Navigation, Ausliefern, Türe öffnen
  - ➔ RL mit temporär erweiterbaren Aktionen
- Integration von zusätzlichem Wissen
- Zustands-/Aktionsabstraktion
  - Zustände mit weniger Zustandsvariablen
  - verschiedene Umweltzustände → auf einen abstrakten Zustand abbilden
  - Unterschiedlich abstrakte Zustände in verschiedenen Makro-Aktionen
  - ➔ Lernen beschleunigen

Options [Sutton, Precup, Singh'99] →Teilaktion

- Beispiel: Roboterbewegung
  - Option : $Move_e$ until end of hallway
  - Start : Irgendein Zustand im Gang
  - Execute: policy, z.B. Fahren im Gang
  - Terminate: wenn s das Ende des Ganges ist
- Lernen = Passender Formalismus gesucht

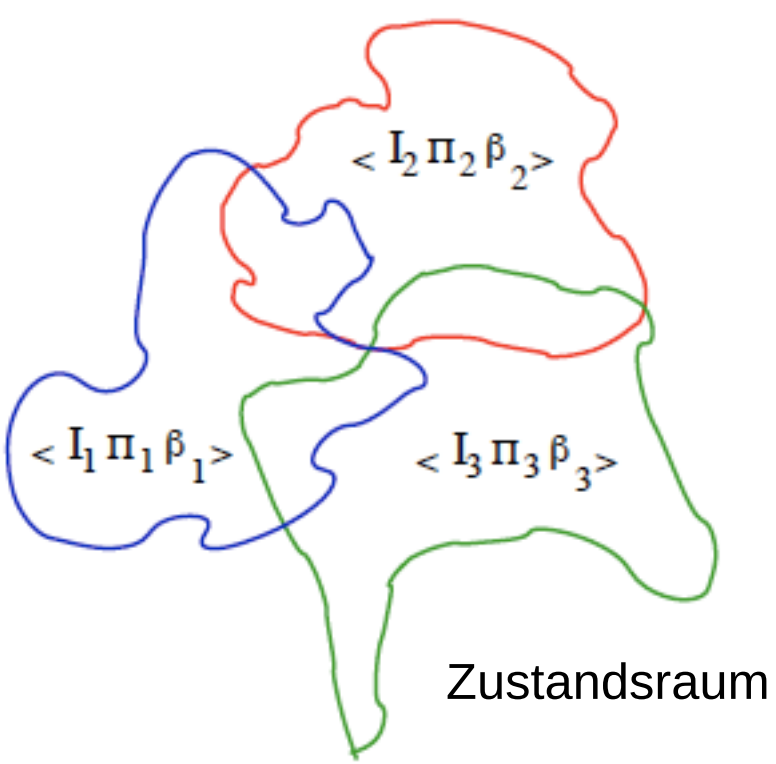


Options [Sutton, Precup, Singh'99]

- Ansatz: Verwende s.g. *Options* = wohl definiertes Verhalten

$$o := (I_o, \pi_o, \beta_o)$$

- I_o : Menge von Zuständen in denen o gestartet werden kann
- $\pi_o(s) : S \rightarrow A^*$
  Policy, während o ausgeführt wird
  - A* kann ebenfalls über eine Policy über weitere Optionen definiert sein
- $\beta_o(s)$: Wahrscheinlichkeit dass o in s beendet wird
- Auf Options o kann eine weitere MDP aufgesetzt werden:
  Policy μ



Lernen auf Options

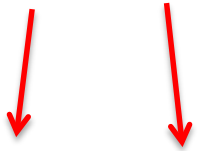
- Eine Option o ist eine zeitlich erweiterte Aktion mit wohl definierter interner policy
- Ein-Schritt-Options sind bisherige Aktionen (die in jedem Zustand zulässig sind und danach enden)
- Die Menge der Options O ersetzt die Menge der Aktionen
- Lernen (RL) kann auf Options stattfinden

  ➔Gelernte Policy $\mu: S \rightarrow O$
- ABER: Lernen auf Optionen erfordert die Erweiterung auf Semi-MDP

Semi - MDP

- Im „klassischen“ MDP spielt nur die sequentielle Natur der Entscheidung eine Rolle, nicht die Zeit, die zwischen zwei Entscheidungen liegt (= Länge einer Aktion)

- Eine Verallgemeinerung ist der $Semi-MDP$
  - Gegeben: T – Dauer einer Aktion, dann betrachtet man für den Zustandsübergang die Wahrscheinlichkeit
    $$P(s', T | s, a)$$
  - Und die Bellmann Gleichungen:



$$Q(s,a) = r(s,a) + \sum_{s',T} \gamma^T P(s', T | s, a) V^*(s')$$

$$= r(s,a) + \sum_{s',T} \gamma^T P(s', T | s, a) \max_{a'} Q(s', a')$$

Semi – MDP – Lernschritt (vgl. Sarsa)

- MDP – Sarsa:

$$Q_{k+1}(s,a) = (1-\alpha_k)Q_k(s,a) + \alpha_k\left[r + \gamma Q_k(s',a')\right]$$

- Semi – MDP:
  Wenn a eine zusammengesetzte Aktion ist und a wird in s ausgeführt, zum Zeitpunkt t , die Transition benötigt τ und man erhält die Teilrewards r_{t+i}

$$Q_{k+1}(s,a)$$
$$= (1-\alpha_k)Q_k(s,a) + \alpha_k\left[r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{\tau-1} r_{t+\tau} + \gamma^{\tau} \max_{a' \in \mathcal{A}_{s'}} Q_k(s',a')\right]$$

Options – nichtdeterministischer Semi-MDP

- Es gilt:
  - R und ein P (keine Wahrscheinlichkeit) lassen sich neu definieren
  - damit kann man RL auf Optionen durchführen

$$R(s,o) = E\{r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{\tau-1} r_{t+\tau} | \mathcal{E}(o,s,t)\}$$

$$P(s'|s,o) = \sum_{\tau=1}^{\infty} p(s',\tau)\gamma^{\tau}$$

$$V_{\mathcal{O}}^{*}(s) = \max_{o \in \mathcal{O}_s} [R(s,o) + \sum_{s'} P(s'|s,o) V_{\mathcal{O}}^{*}(s')]$$

$$Q_{\mathcal{O}}^{*}(s,o) = R(s,o) + \sum_{s'} P(s'|s,o) \max_{o' \in \mathcal{O}_{s'}} Q_{\mathcal{O}}^{*}(s',o')$$

- Lernen auf Options bleibt gleich (r – akkumulierter Reward nach o)

$$Q_{k+1}(s,o) = (1-\alpha_k) Q_k(s,o) + \alpha_k \left[r + \gamma^{\tau} \max_{o' \in \mathcal{O}_{s'}} Q_k(s',o') \right]$$

Diskussion Options

- Der Options-Ansatz strukturiert: S, A, π, G
- Lernen, findet zunächst auf Makro-Ebene statt und wird vereinfacht
  ← lediglich Start- und Terminalzustände sind für Options relevant

  - Hintergrundwissen wird (derzeit) benötigt um sinnvolle Options und primitive Aktionen zu definieren
  - Option-Policies werden meist vorgegeben

- „Intra-Option – Lernen“ kann für Optionen verwendet werden
  ➔ Reduzierung des Lernens zunächst auf eine Untermenge der Zustände und Aktionen (z.B. Bewegung im Gang)
  ➔ Automatische Erkennung von Zwischenzielen ist Gegenstand aktueller Forschung
    - Ein Ansatz sind z.B. häufig erreichte Zustände oder Regionen die häufig „durchwandert werden“

Inhalt

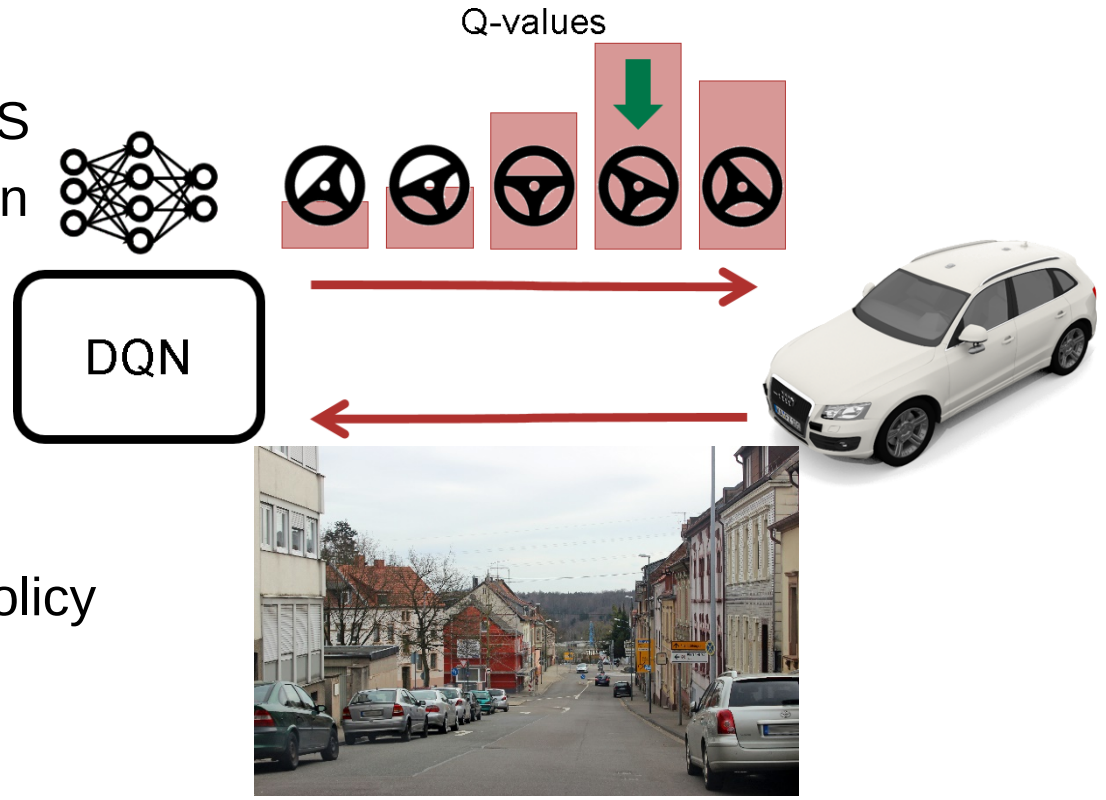
- Wdh. Was ist RL?
- Problematik RL

- Erweiterte Methoden
  - Effiziente Funktionsapproximation
  - Hierarchisches Reinforcement Learning
    - Options

- Deep Reinforcement Learning
  - Deep Q-Netzwerke

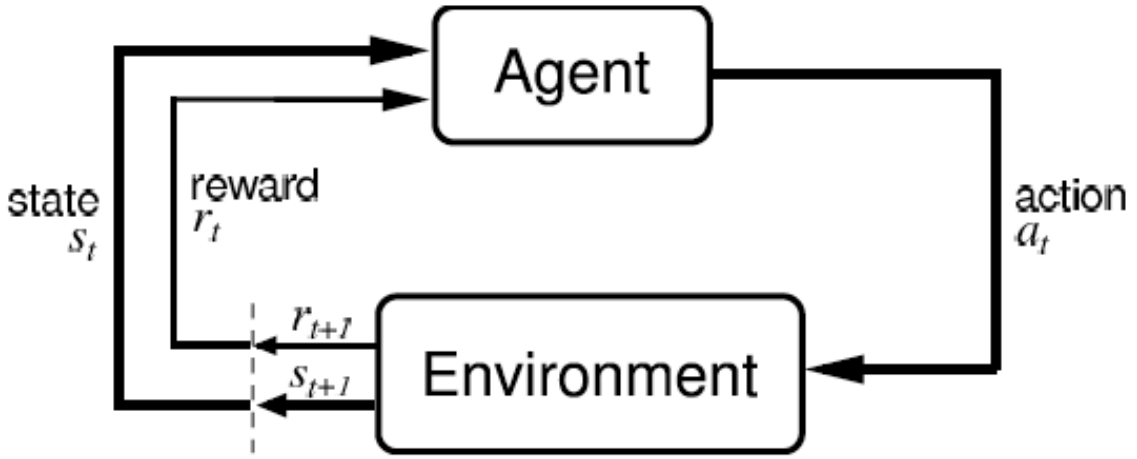
Deep Reinforcement Learning

- Deep Q-Learning [Mnih 2013]
  - verwendet CNN um effizient Q-Funktion zu approximieren
  - arbeitet mit hochdimensionalem Zustandsraum S
  - gute Performance in unterschiedlichen Domänen
- Schwierigkeiten
  - großer Datenbedarf
  - hohe Korrelation der Daten
  - beobachtete Daten beeinflusst durch aktuelle Policy
  - ➔ *Experience Replay, Sollwert-Netzwerk*



Experience Replay

- Idee: Speichern und Sampling von Übergängen (s_t, a_t, r_t, s_{t+1})
  - Übergänge können mehrfach genutzt werden
  - Korrelation der Beobachtungen wird gebrochen
  - verhindert Feedback-Loop



Training mit Experience Replay

[Mnih 2013]

- Initialisiere Deep Q-Netzwerk (DQN) mit zufälligen Gewichten θ
- Iteriere über Trainingsepisoden
  - Wähle Aktion $a_t = argmax_a(Q(s_t, a;\ \theta))$ (ε-greedy)
  - Führe a_t aus und erhalte Reward r_t und Zustand s_{t+1}
  - Speicher Übergang (s_t, a_t, r_t, s_{t+1}) in M
  - Sample zufälligen Batch von Übergängen (s_i, a_i, r_i, s_{i+1})
    - Definiere Sollwert $y_i = \begin{cases} r_i, & falls\ s_{i+1}\ terminal \\ r_i + \gamma\ max_a\big(Q(s_{i+1}, a;\ \theta)\big), & sonst \end{cases}$
    - Gradientenabstieg mit Error $(y_i - Q(s_i, a_i;\ \theta))^2$ auf Q-Netzwerk

Gleiches Netz zur Ist- und Sollwertbestimmung!

- Bewertung der Zustände nicht unabhängig voneinander
- kann zu Oszillation oder Divergenz der Policy führen

Verwendung eines Sollwert-Netzwerks

[Mnih 2015]

- Initialisiere DQN mit zufälligen Gewichten θ
- Initialisiere Sollwert-Netzwerk $\hat{Q}$ mit $\hat{\theta} = \theta$
- Iteriere über Trainingsepisoden
  - Wähle Aktion $a_t = argmax_a(Q(s_t, a;\, \theta))$ (ε-greedy)
  - Führe a_t aus und erhalte Reward r_t und Zustand s_{t+1}
  - Speicher Übergang (s_t, a_t, r_t, s_{t+1}) in M
  - Sample zufälligen Batch von Übergängen (s_i, a_i, r_i, s_{i+1})
    - Definiere Sollwert $y_i = \begin{cases} r_i, & falls\ s_{i+1}\ terminal \\ r_i + \gamma\, max_a\left(\hat{Q}(s_{i+1}, a;\, \hat{\theta})\right), & sonst \end{cases}$
    - Gradientenabstieg mit Error $(y_i - Q(s_i, a_i;\, \theta))^2$ auf Q-Netzwerk
    - Alle n Schritte: Setze $\hat{Q} = (1 - \alpha)\hat{Q} + \alpha Q$

Anwendungsbeispiele

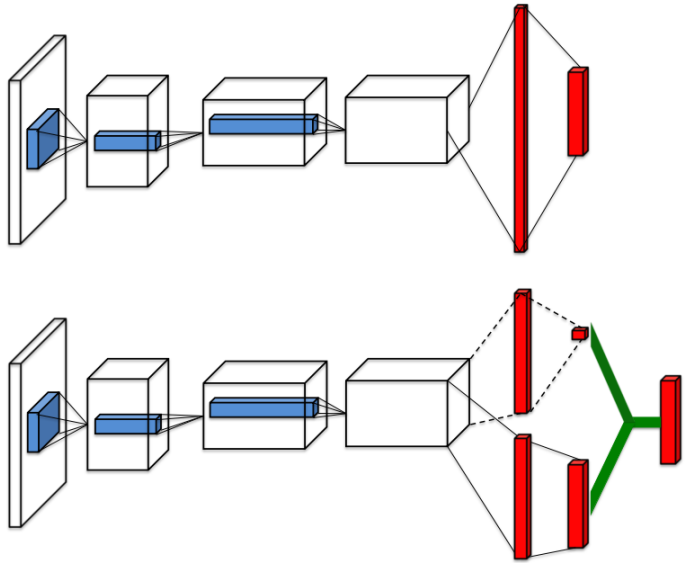
[Mnih 2015]

Erweiterungen des Deep Q-Learning

- Priorisiertes Experience Replay [Schaul2016]
  - behalte wertvolle Übergänge länger im Speicher
- Actor – Critic Ansatz [Silver2014,Lillicrap2016]
  - Deep Deterministic Policy Gradient (DDPG)
  - Actor $\mu(s;\ \theta^{\mu})$
    - bildet Zustand s auf eine Aktion a ab
    - spezifiziert die Policy π
  - Critic $\mathrm{Q}(s, a;\ \theta^{Q})$
  - trennt Lernen der Q-Funktion und Lernen der Policy
  - ermöglicht kontinuierlichen Aktionsraum
- Asynchrone Agenten
  - Dekorrelation der Übergänge durch mehrere Agenten
  - ermöglicht On-Policy Lernverfahren

Erweiterungen des Deep Q-Learning

- Double Q-Learning in DQNs [vanHasselt2016]
  - Auftrennung des max-Operators in Aktionsauswahl und -bewertung
  - Reduzierung von überschätzten Erwartungswerten
  - $y_i = r_i + \gamma \hat{Q}\left(s_{i+1}, argmax_a Q(s_{i+1}, a;\, \theta);\, \hat{\theta}\right)$
- Dueling DQN [Wang2016]
  - Advantage: $\mathrm{A}(\mathrm{s},\mathrm{a}) = \mathrm{Q}(\mathrm{s},\mathrm{a}) - \mathrm{V}(\mathrm{s})$
    - oder: $\mathrm{Q}(\mathrm{s},\mathrm{a}) = V(s) + A(s,a)$
  - Schätze $\mathrm{V}(\mathrm{s})$ und $\mathrm{A}(\mathrm{s},\mathrm{a})$ um $\mathrm{Q}(\mathrm{s},\mathrm{a})$ zu approximieren
  - bessere Generalisierung über Aktionsraum



[Wang2016]

Thinking Big…

"... consider maze domains. Reinforcement learning researchers, including this author, have spent countless years of research solving a solved problem! Navigating in grid worlds, even with stochastic dynamics, has been far from rocket science since the advent of search techniques such as A*."

☺ -- David Andre

Literatur

Siehe ML I ← RL

A.G. Barto and S. Mahadevan: Recent Advances in Hierarchical Reinforcement Learning. 2003

Weiterführende Literatur:

D. Ormoneit and S. Sen Kernel-Based Reinforcement Learning. Journal of Machine Learning, 2002

M.H. Lagoudakis and R. Parr Least-Squares Policy Iteration. Journal of Machine Learning Research, 2003

M. Riedmiller: Neural Fitted Q Iteration. Proceedings of the 17th European Conference on Machine Learning (ECML), 2005

Weitere Hierarchische RL - Ansätze

ANHANG