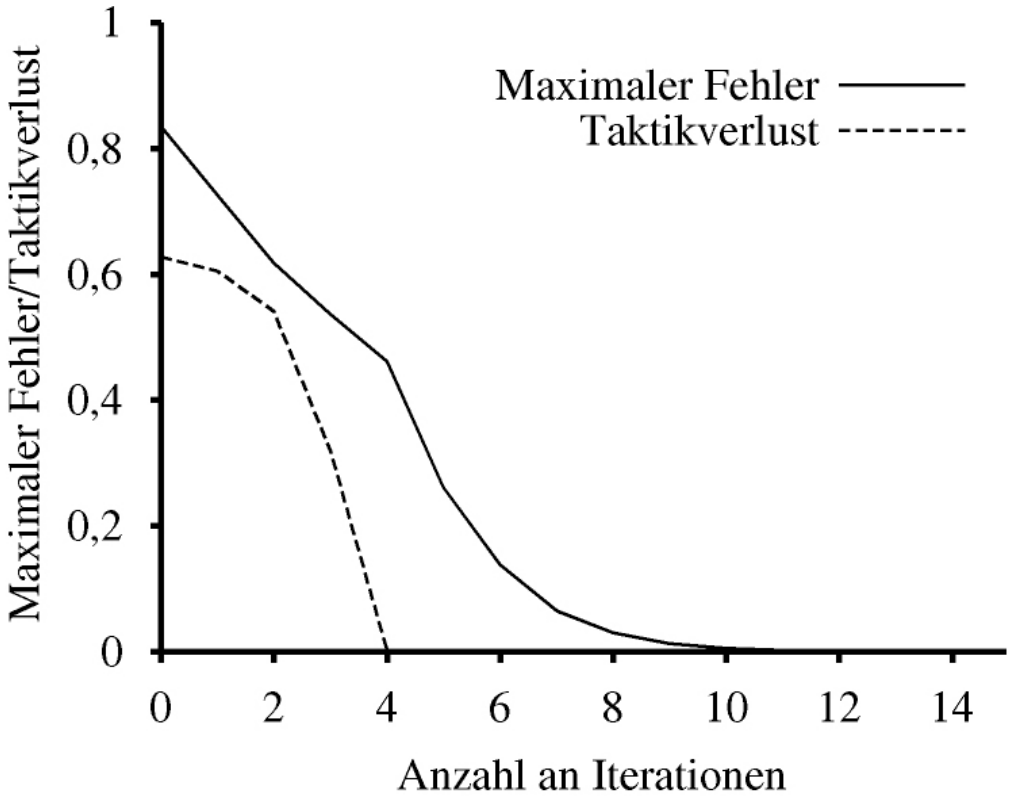
Ist das wirklich schlimm?

Gesucht ist nicht der „wahre“, objektive Nutzen eines Zustands, sondern die optimale *Policy*!



Die optimale *Policy* ergibt sich zumeist schon bei recht grob approximierten Utilities!

➥ ***Policy Iteration***

Optimale *Policies* ohne präzise *Utilities*

Grundidee der *Policy Iteration*: Starte mit beliebiger (zufällig gewählter) *Policy*, iteriere die folgenden beiden Schritte:

- **Bewertung**: Berechne den Nutzen U_i jedes Zustands unter der aktuellen *Policy* π_i
- **Verbesserung**: Berechne, wenn möglich, basierend auf den aktuellen Nutzen-Werten eine bessere *Policy* π_{i+1}

Effizienter als *Value Iteration*, weil für Bewertung aktueller Nutzen der Zustände nicht über *alle* möglichen Aktionen maximiert werden muss! Vereinfachung der Bellmann-Gleichung (n lineare Gleich. mit n Unbekannten $U_i(s)$ bei n Zuständen, lösbar in $O(n^3)$):

$$U_i(s) = R(s) + \gamma \sum_{s'} T(s,\pi_i(s),s')U_i(s')$$

Policy Iteration

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, transition model T
  local variables: U, U', vectors of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_a \sum_{s'} T(s,a,s')\ U[s'] > \sum_{s'} T(s,\pi[s],s')\ U[s']$ **then**
        $\pi[s] \leftarrow \text{argmax}_a \sum_{s'} T(s,a,s')\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

POLICY-EVALUATION ist die Bewertung auf der vorigen Folie

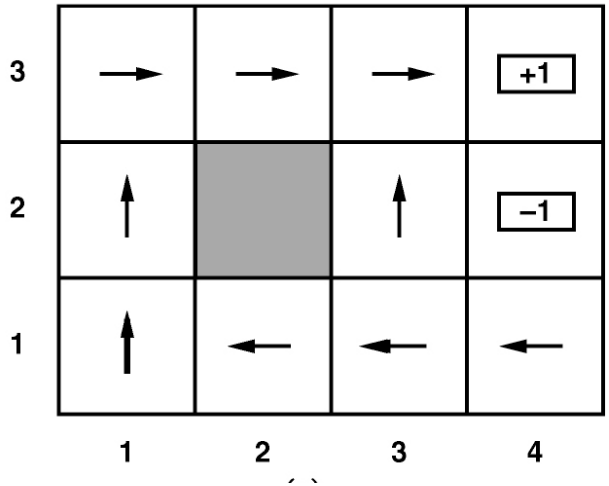
Und was macht man, wenn das MDP nicht bekannt ist?

Dafür gibts Reinforcement-Lernen!

Russell/Norvig Kap. 21
Ertel Kap. 10

Erster Schritt: Passives RL

- **Gegeben**: Beobachtetes Verhalten
  (unbekanntes MDP mit unbekannter *Policy* $\pi(s)$)
- **Finde/lerne**: Nutzenfunktion $U^{\pi}(s)$
  (verbesserte, präzisere, aktuellere Version)
- „Passiv“, weil Aktionen aus unbekanntem $\pi(s)$ nur beobachtet werden



Was wir haben:

- Die Def. der Nutzenfunktion unter *Policy* π

$$U^{\pi}(s) := EU\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \middle| \pi, s_0 = s\right]$$

- Beobachtete Aktions/Zustands/*Reward*-Sequenzen, z.B.
  - $(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$
  - $(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$
  - $(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (4,2)_{-1}$

Update-Regel für die Nutzenfunktion

- Die einzig vorhandene Information kommt aus der Beobachtung von Aktionssequenzen und *Rewards*
- Gegeben hinreichend viele Aktionssequenzen, ergibt sich der Nutzen eines Zustands aus dessen *Reward* und anteilig den Nutzen der Nachfolgezustände

Die ***Temporal Difference*** (**TD**) Update-Regel im akt. Zust. s':

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha\big(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s)\big)$$ (s ist d. Vorgänger v. s')

- γ Abschlags-Faktor (s.o.)
- α **Lernrate**: Wie unmittelbar soll e. in Aktionssequenz festgestellte Nutzendifferenz im Update berücksichtigt werden? (α kann von Zahl der Zustandsbesuche abhängen)

TD braucht kein explizites Modell der Umgebung (T,R)!

TD-Lernen

function PASSIVE-TD-AGENT(*percept*) **returns** an action

inputs: *percept*, a percept indicating the current state s' and reward signal r'

static: π, a fixed policy

U, a table of utilities, initially empty

N_s, a table of frequencies for states, initially zero

s, a, r, the previous state, action, and reward, initially null

if s' is new **then** $U[s'] \leftarrow r'$

if s is not null **then do**

increment $N_s[s]$

$U[s] \leftarrow U[s] + \alpha(N_s[s])(r + \gamma U[s'] - U[s])$

if TERMINAL?$[s']$ **then** $s, a, r \leftarrow$ null **else** $s, a, r \leftarrow s', \pi[s'], r'$

return a

Lernrate hängt hier ab von Zahl der Zustandsbesuche

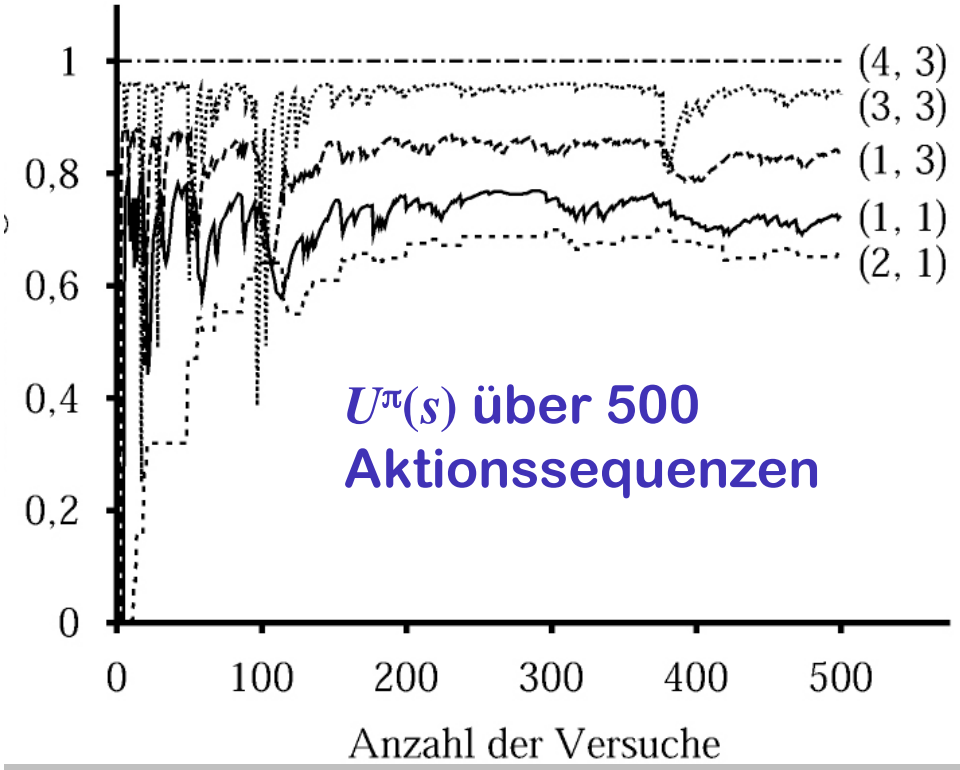
TD-Regel

Struktur dieses Algorithmus (nur 1 Lernschritt) und Ausgabe (Aktion) seltsam formuliert für passives TD-Lernen!

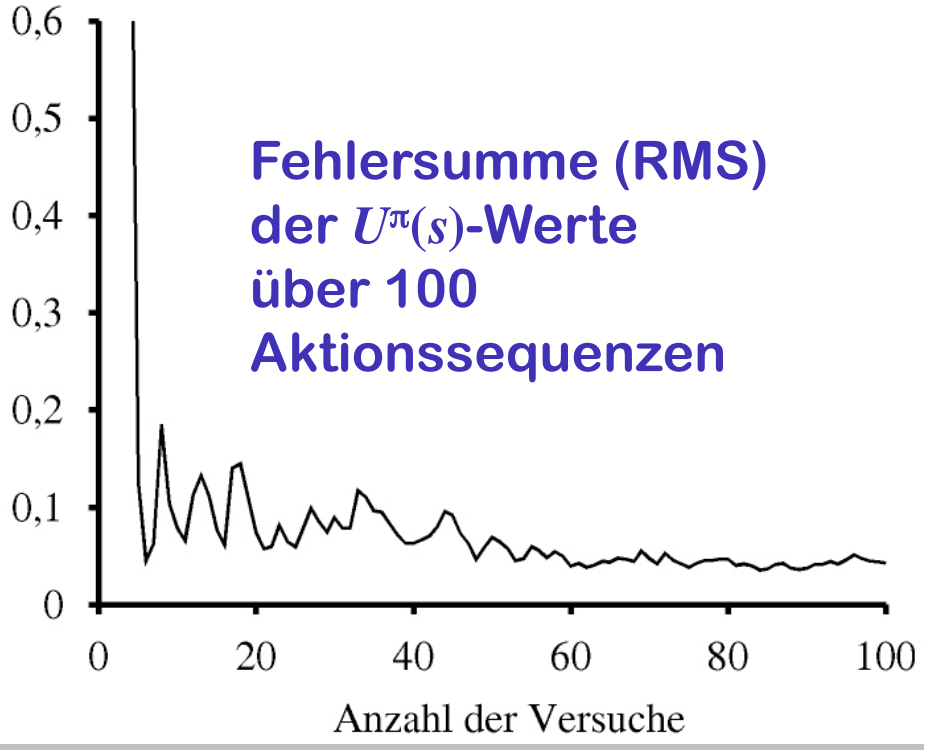
Ergebnisse

Satz: Gemittelt über Aktionssequenzen konvergiert $U^{\pi}(s)$ gegen den korrekten Wert (s. Folie 324)

Typische Ergebnisse (qualitativ)



$U^{\pi}(s)$ über 500 Aktionssequenzen



Fehlersumme (RMS) der $U^{\pi}(s)$-Werte über 100 Aktionssequenzen

Zweiter Schritt: Aktives RL

- **Gegeben**: ein „unbekanntes“ MDP (ohne bekannte Policy)
- **Finde/lerne**: für jeden Zustand s die optimale Aktion a (nicht unbedingt den präzisen Nutzenwert $U(s)$)
- „Aktiv“, weil MDP-Plan nicht vorgegeben ist, sondern gefunden werden muss
- Entspricht Planen ohne Domänenmodell!

Was wir haben:

- Bellmann-Gleichungen $U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$ (Folie 326)
  als Beschreibung eines „Fixpunkts“ von U, R, T
- … wobei wir weder U noch R noch T kennen!
- Beobachtete Aktions/Zustands/Reward-Sequenzen, wie eben

Die Q-Funktion

- Ziel ist, modellfrei optimale Aktionen für Zustände zu lernen (nicht mehr eine Nutzenfunktion für gegebenes π)
- Ersetze Nutzen eines Zustands $U(s)$ durch Nutzen einer Aktion im Zustand: $Q(a,s)$, wobei $U(s) = \max_a Q(a,s)$
- entsprechend Bellmann-Gleichung in Q

  (formuliert nach wie vor Fixpunkt der Funktionswerte):

$$Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_b Q(b,s')$$

- TD-Update-Regel in Q-Version

  (vgl.: $U^\pi(s) \leftarrow U^\pi(s) + \alpha(R(s) + \gamma U^\pi(s') - U^\pi(s))$

$$Q(a,s) \leftarrow Q(a,s) + \alpha\left(R(s) + \gamma \max_b Q(b,s') - Q(a,s)\right)$$

  (α kann von der Frequenz der Zustandsbesuche abhängen)

Wissen ausbauen oder ausbeuten?

… Ertel: Erkunden oder verwerten? … Englisch: *exploration vs. exploitation*

- „Mittendrin“ im Lernen
  haben wir approximative Nutzenwerte, Aktionsmodelle
- Sollen wir dann schon „gut“ handeln, müssten wir immer die dann optimale Aktion wählen – gemäß dem, was wir dann wissen („Ausbeuten“ des aktuell Gelernten)
- Gäbe es eine bessere Aktion, fänden wir sie nie

↪ Um das zu tun, müssen wir „manchmal“ gegen das aktuell bekannte Optimum agieren, um möglicherweise Besseres zu finden („Ausbauen“ des aktuell Gelernten)

Explorationsfunktion

$$f(u,n) = \begin{cases} R^+ \text{ falls } n < N \\ u \text{ sonst} \end{cases}$$

- u Nutzen- bzw. q-Wert
- n Häufigkeit, wie oft Zustand besucht
- N feste Schranke
- R^+ feste Schätzung eines max. Rewards

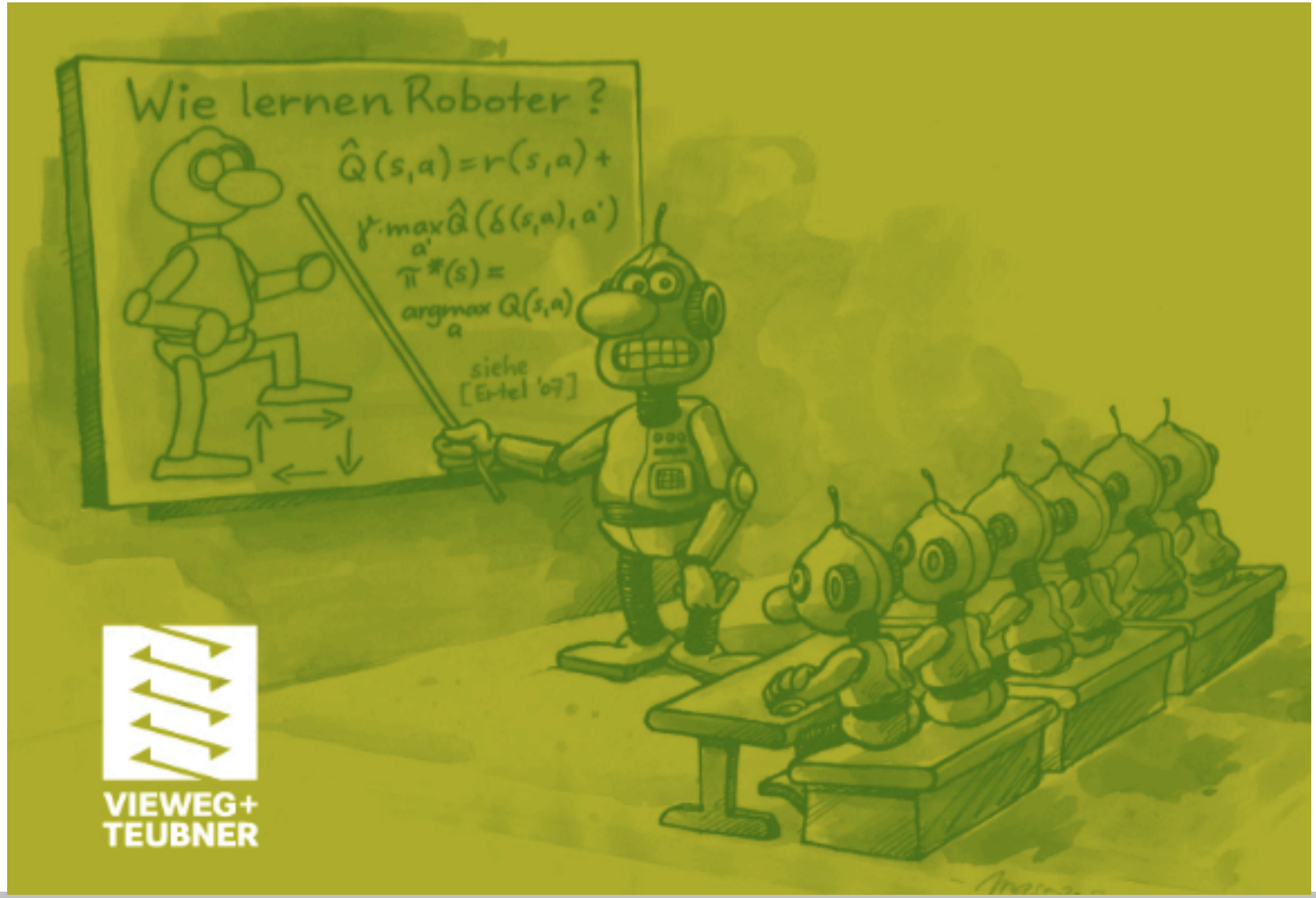
Q-Lernen

function Q-LEARNING-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  static: Q, a table of action values index by state and action
    N_{sa}, a table of frequencies for state-action pairs
    s, a, r, the previous state, action, and reward, initially null

  if s is not null **then do**
    increment $N_{sa}[s, a]$
    $Q[a, s] \leftarrow Q[a, s] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[a', s'] - Q[a, s])$
  if TERMINAL?$[s']$ **then** $s, a, r \leftarrow$ null
  else $s, a, r \leftarrow s', \text{argmax}_{a'} f(Q[a', s'], N_{sa}[a', s']), r'$
  return a

Geeignete Parameter der f-Fkt. vorausgesetzt, konvergiert die Funktion in eine optimale *Policy*

Q-Lernen in der Robotik



Weiterführendes zum RL

- Will man modellfrei sein, oder will man eigentlich (auch) das Umgebungsmodell haben?
- Wie integriere Vorwissen über optimales/gutes Handeln?
- Wie kommt man zurecht mit Veränderung in der Umgebung? Muss man
  - erst alles Gelernte „abtrainieren“ und dann das Neue lernen
  - oder kann man Teile des früher Gelernten übernehmen?

Fazit Lernen

Erinnerung Folie 227

Definition 8.1 *Ein Agent heißt lernfähig, wenn sich seine Leistungsfähigkeit auf neuen, unbekannten Daten, im Laufe der Zeit (nachdem er viele Trainingsbeispiele gesehen hat) verbessert (gemessen auf einem geeigneten Maßstab).*

- Hübsche Definition, charakterisiert aber nicht gut die Basis-Lernverfahren, die wir hier hatten
- Der „Maßstab“ ist praktisch immer Reproduktion der Trainingsmenge (überwachte Verfahren, ILP) bzw. „kleinster Abstand zu Zielfunktion“ (unüberwacht, Reinforcement) – bei Agenten ohne eigenen Zweck geht das nicht anders
- *Overfitting* ist dann ein Riesenproblem
- Lernen wird heftig eingesetzt in Data Mining und Robotik
- KI-Systeme ohne Lernen sind starr;
  Lernverfahren ohne Systemkontext („Zweck“) sind witzlos