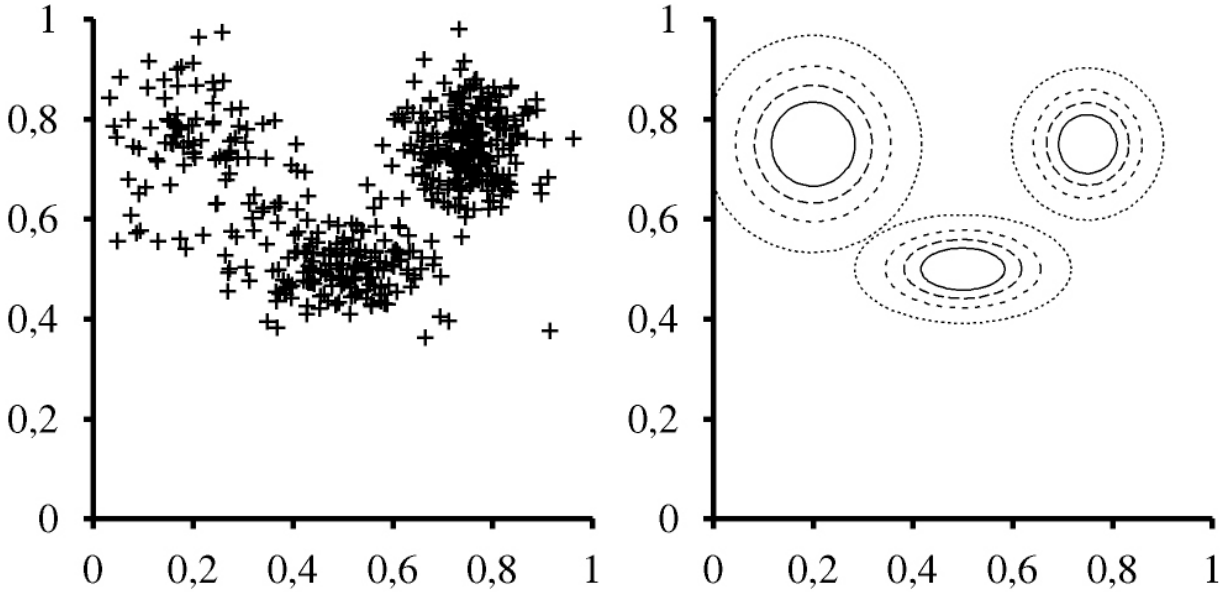
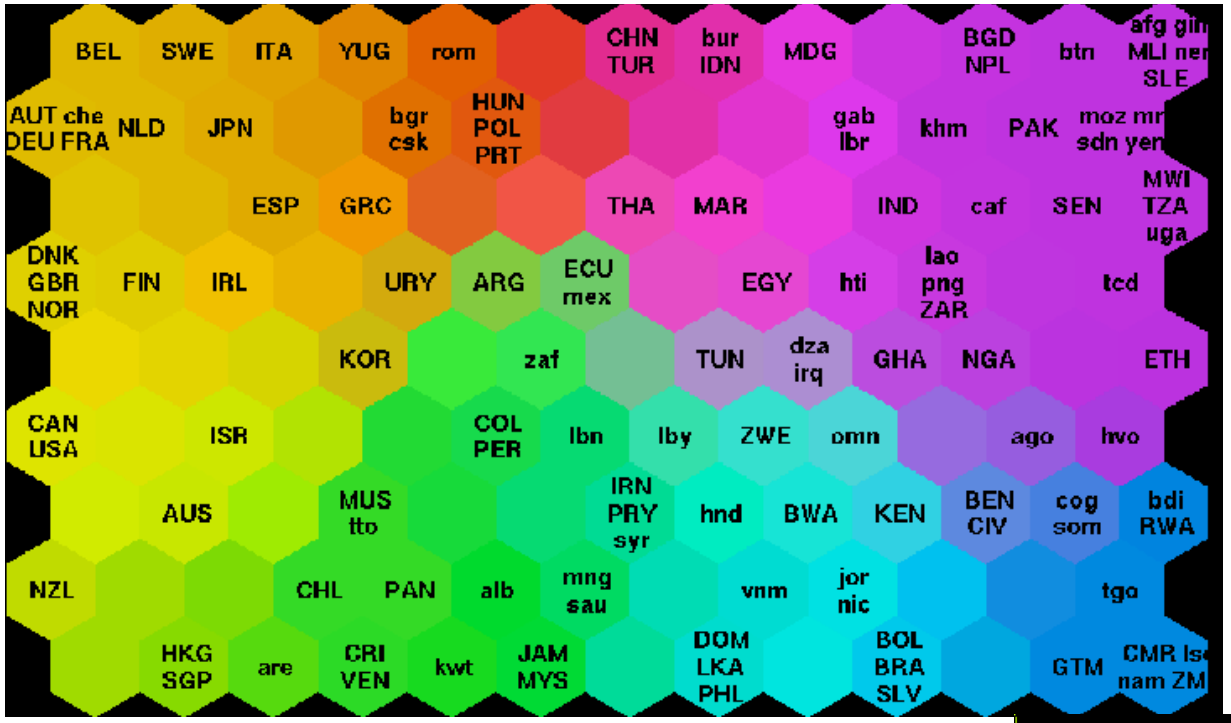
Unüberwachtes Lernen

- Keine ausdrücklichen Lernbeispiele
- Zum Lernen verwende die Information, die „von allein“ da ist: Merkmalsvektoren von Objekten; Logikformeln; …
- Ziel ist die automatische Strukturierung: Cluster bzgl. Merkmalen; Definitionen bzgl. Prädikaten; …



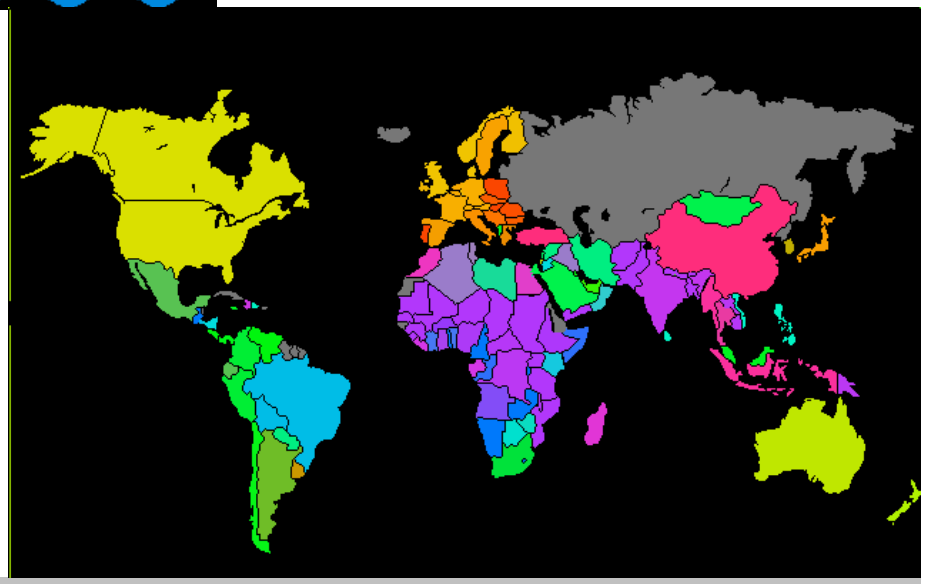
Ergebnis von unüberwachtem Clustering, Russell/Norvig Fig. 20.8



Beispiel für Clustering: Kohonen-Karten

Übertragung auf geographische Verteilung

- 39 Faktoren für Lebensstandard (Weltbank, 1992: Einkommen, Infrastruktur, Gesundheit, …)
- gewichte Faktoren; Abbildung in 2-dim. Cluster; Nachbarschaft in Farbcodierung



http://www.cis.hut.fi/research/som-research/worldmap.html

Lernen in Logik

Beispiel: Aus Lernbeispielen (und evtl. Weiterem) erzeuge z.B.

$$\begin{aligned}\forall r.\ Warten(r) \Leftrightarrow \quad & Gäste(r,Einige) \\ \vee\ & [Gäste(r,Voll) \wedge Hungrig(r) \wedge Typ(r,Französisch)] \\ \vee\ & [Gäste(r,Voll) \wedge Hungrig(r) \wedge Typ(r,Thai) \wedge Frei/Sams(r)] \\ \vee\ & [Gäste(r,Voll) \wedge Hungrig(r) \wedge Typ(r,Burger)]\end{aligned}$$

… und bevorzuge „einfache“ Formeln (*Ockham's Razor*)!

Vorteile

- Überwachtes Lernen (wie mit DTL) ist echte Untermenge davon
- Lerne beliebige Prädikate/Relationen (nicht nur 1-stellige Attribute)
- Kann Regeln induzieren *ohne* explizite Lernbeispiele
- Beziehe „Hintergrundwissen“ ein bzw. baue es aus

(Potenzieller) Nachteil

- Größere Ausdrucksfähigkeit macht Lernen komplexer

Erscheinungsformen Logikbasierten Lernens

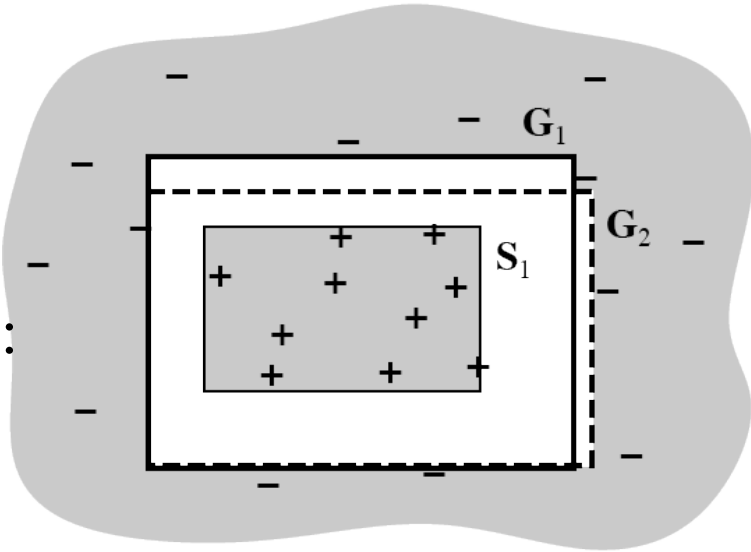
- **Erklärungsbasiertes** Lernen (**EBL**): *Deduziere* Klassifikation der Lernbeispiele als Instanzen allgemeiner Prinzipien (Lernen spezifischer „Abkürzungen“ von Deduktion)
- **Relevanzbasiertes** Lernen (**RBL**): Deduziere klassifikations-relevante Information, dann Hypothesen
- **Wissensbasiertes induktives** Lernen (**KBIL**):

  $$Background \wedge Hypothese \wedge Lernbeispiele \models Klassifikation$$

  - Hypothese ist zu finden ➥ **induktives** Lernverfahren
  - Hypoth. muss konsistent sein mit $Background \wedge Lernbeispiele$

Hier: **Induktives Logisches Programmieren** (**ILP**)

Intuition: Der Versionenraum



Gegeben: log. Theorie (Formelmenge) T, sodass $T \models P(a,b)$ und $T \models \neg P(b,b)$, und keine weitere Folgerbarkeit bzgl. P. Also:

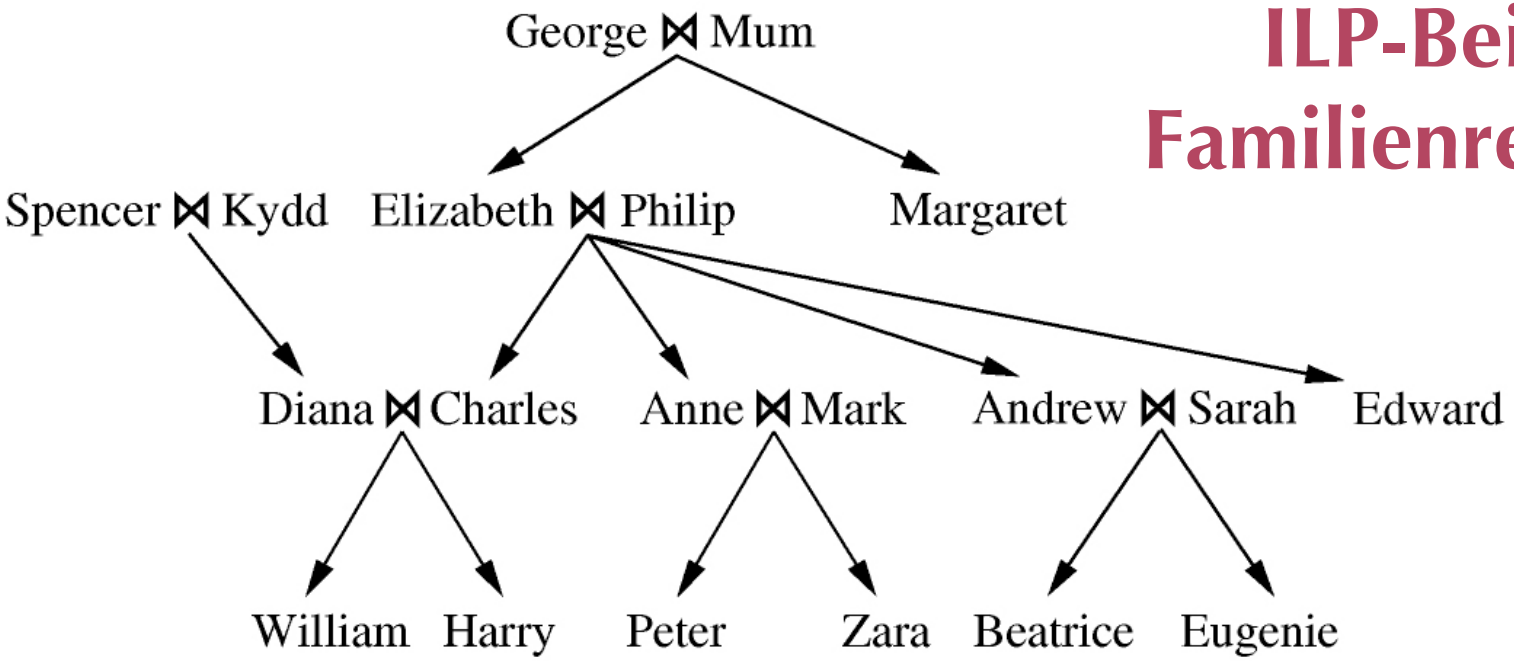
- für (a,b) ist P wahr,
- für (b,b) ist P falsch,
- für (a,a), (b,a) ist P ungewiss (kann wahr oder auch falsch sein)

Soll die **Extension** (Menge der erfüllenden Argument-Paare) von P präzise charakterisiert werden, ist

- $\{(a,b)\}$ oder $\{(b,b)\}$ zu **generalisieren** (Beispiele hinzu)
- Gegenrichtung: **Spezialisieren** (falsch Klassifizierte weg)

In Logik verändert man die Extension eines Prädikats durch Hinzufügen von Formeln zur Theorie zu T

ILP-Beispiel: Familienrelationen



Gegeben:

- Vollständige Repräsentation des Stammbaums mittels Prädikaten *Vater*, *Mutter*, *Paar*, *Mann*, *Frau*
- einige der 20x20 Instanzen (pos., neg.) des Prädikats *Opa*(*x*,*y*)

Gesucht:

- z.B. Definition von *Opa*(*x*,*y*) in Termini der anderen Prädikate
- Form der Def.: **Disjunktion von Hornklauseln**

Beispiel: Opadefinitionskandidaten

Gegeben: $Opa(George, Anne)$, $Opa(Philip, Peter)$, $Opa(Spencer, Harry)$,
$\neg Opa(Anne, Anne)$, $\neg Opa(Harry, Zarah)$, $\neg Opa(Charles, Philip)$

Kandidaten für Definitionen („PROLOG-artige“ Notation, aber „⇒“)

- $\Rightarrow Opa(x,y)$.
  Falsch für alle Gegenbeispiele ➥ spezialisieren (durch „Raten“)!
  Zufällige Kandidaten:
  - $Vater(x,y) \Rightarrow Opa(x,y)$. (Falsch für alle Beispiele)
  - $Paar(x,z) \Rightarrow Opa(x,y)$. (Falsch für einige Gegenbeispiele)
  - $Vater(x,z) \Rightarrow Opa(x,y)$. (Falsch für mehr Gegenbeispiele)
  - … wähle #2 zum Spezialisieren etc.

… bis Klauseln gefunden, die alle Positiv-, keine Negativbeispiele implizieren ➥ deren Disjunktion ist die Definition!

FOIL (*First Order Inductive Learner*)

```
function FOIL(examples, target) returns a set of Horn clauses
   inputs: examples, set of examples
           target, a literal for the goal predicate
   local variables: clauses, set of clauses, initially empty

   while examples contains positive examples do
       clause ← NEW-CLAUSE(examples, target)
       remove examples covered by clause from examples
       add clause to clauses
   return clauses
```

- Potenziell jede Klausel der Sprache für NEW-CLAUSE möglich
- Heuristiken/Bedingungen zu Klauseln: Variablen, Länge, …
- Mehr bei Russell/Norvig, Kap. 19.5

Inverse Resolution

- Der Suchraum beim „Klauselraten“ à la FOIL ist riesig
- Wenn das Gelernte die Form

  $$\mathit{Background} \wedge \mathit{Hypothese} \wedge \mathit{Lernbeispiele} \models \mathit{Klassifikation}$$

  haben soll, muss

  $$\mathit{Background} \wedge \mathit{Hypothese} \wedge \mathit{Lernbeispiele} \wedge \neg\mathit{Klassifikation}$$

  durch Resolution widerlegbar sein
- Könnte man dann nicht statt Klauselraten gezielt solche Klauseln als *Hypothese* suchen, die für die Widerlegung von

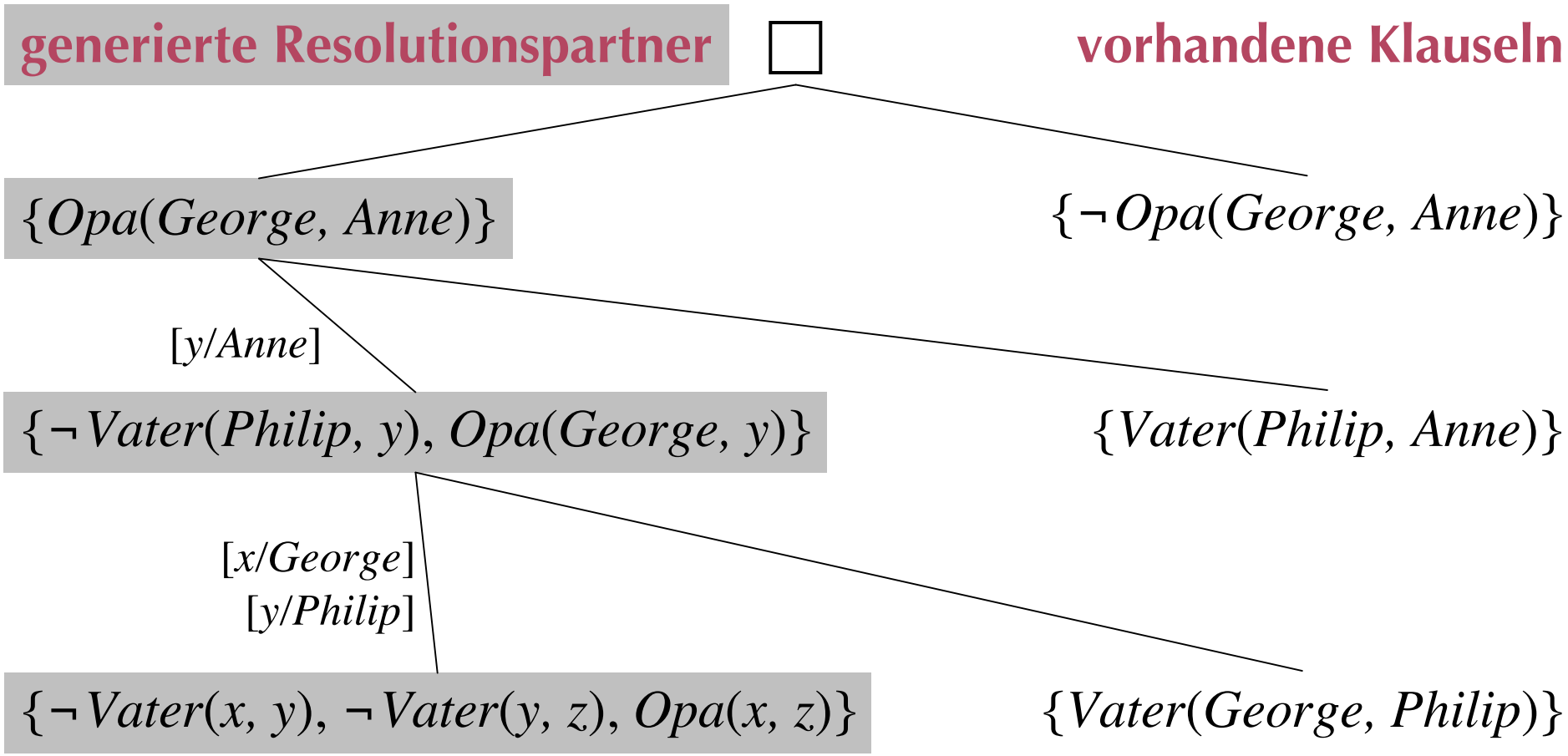
  $$\mathit{Background} \wedge \mathit{Lernbeispiele} \wedge \neg\mathit{Klassifikation}$$

  fehlen?

➥ **Inverse Resolution**

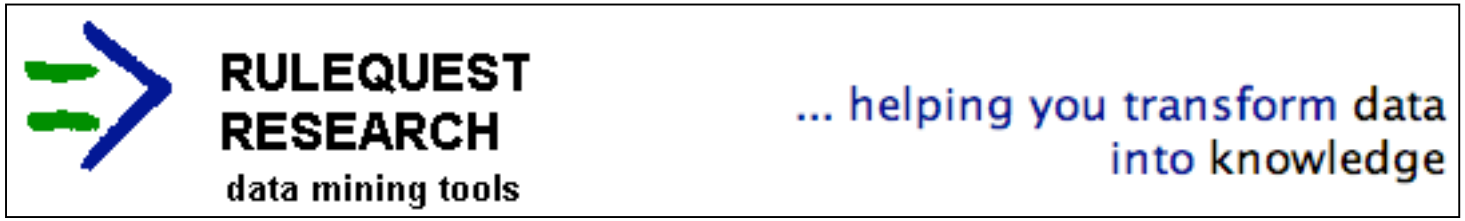
Beispiel für Inverse Resolution

Wir wollen das Positivbeispiel *Opa*(*George, Anne*) widerlegen



ILP und die Praxis

- Um Klauselraten/FOIL oder Inverse Resolution praxistauglich zu machen, gibt es viel an Theorie und Heuristiken
- Es ist sogar möglich, *neue,* Sinn tragende Prädikate zu „entdecken“, welche die Klauselmenge „kompakter“ machen
- Mehr bei Russell/Norvig, Kap. 19.5
- ILP-Systeme werden praktisch beim *data mining* eingesetzt



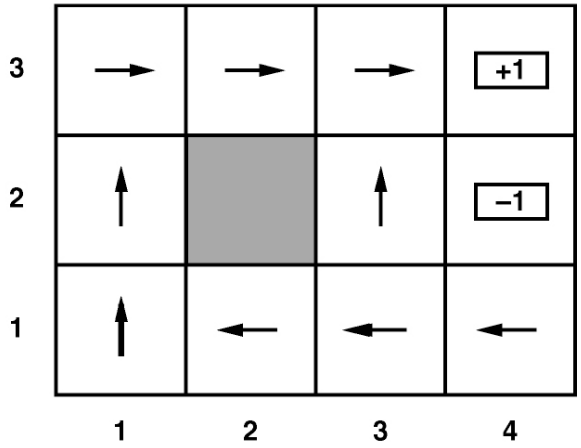
Logo der Firma von Ross Quinlan, Entwickler von FOIL

Reinforcement-Lernen (RL)

- **Ziel**: Lerne auf Grund von Reward/Reinforcement-Signalen aus der Umgebung optimales Handeln in sequenziellen Entscheidungsproblemen
- Ähnlich zu PO/MDPs in Kapitel 5.3
  (viele Gleichungen von dort gelten auch hier), **aber**:
  - keine Kenntnis des Transitionsmodells T
  - keine Kenntnis der Reward-Funktion R
  - während beim Entscheiden unter Unsicherheit eine optimale Entscheidung (MDP-Plan) „objektiv gegeben“ ist (man muss sie nur ausrechnen) …
  - … muss nun zunächst/implizit das Umgebungsmodell „erlernt“ werden

Erster Schritt: Passives RL

- **Gegeben**: MDP mit einem MDP-Plan $\pi(s)$
- **Finde/lerne**: Nutzenfunktion $U^{\pi}(s)$
- „Passiv“, weil Aktionen aus $\pi(s)$ einfach nur ausgeführt werden



Beispiel wie in Kap.5.3

Was wir haben:

- Die Def. der Nutzenfunktion $U^{\pi}(s) := E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \middle| \pi, s_0 = s\right]$ (wie Kap. 5.3, Folie 385)
- Beobachtete Aktions/Zustands/Reward-Sequenzen, z.B.
  - $(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3) \rightarrow (4,3)_{+1}$
  - $(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (3,3) \rightarrow (4,3)_{+1}$
  - $(1,1)_{-.04} \rightarrow (2,1)_{-.04} \rightarrow (3,1)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (4,2)_{-1}$

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

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha\big(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s)\big)$$

- γ Abschlags-Faktor (wie in Kap. 5.3)
- α **Lernrate**: Wie unmittelbar soll eine bei einer Aktionssequenz festgestellte Nutzen-Differenz im Update berücksichtigt werden? (α kann von der Frequenz der Zustandsbesuche abhängen)

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

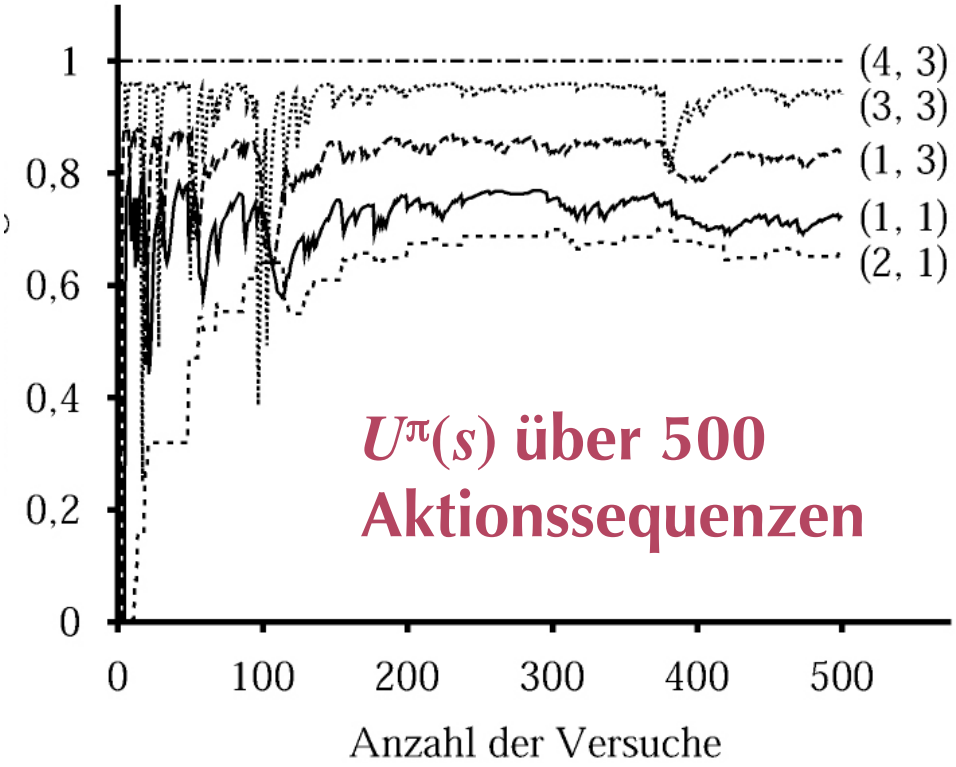
TD-Lernen

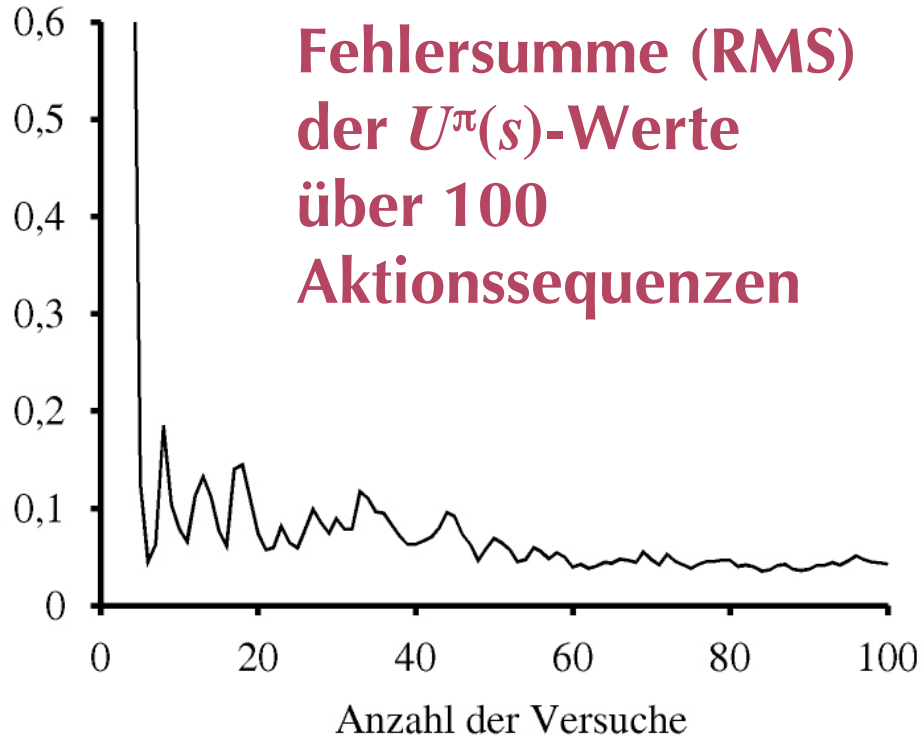
function PASSIVE-TD-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  static: π, a fixed policy
    U, a table of utilities, initially empty
    N_s, a table of frequencies for states, initially zero
    s, a, r, the previous 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

Ergebnisse

Satz: Gemittelt über Aktionssequenzen konvergiert $U^{\pi}(s)$ gegen den korrekten Wert (s. Folie 432)





Zweiter Schritt: Aktives RL

- **Gegeben**: ein „unbekanntes“ MDP
- **Finde/lerne**: für jeden Zustand s die optimale Aktion a
- „Aktiv“, weil MDP-Plan nicht vorgegeben ist, sondern gefunden werden muss

Was wir haben:

- Bellmann-Gleichungen $U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$
  (wie Kap. 5.3, Folie 386)
  als Beschreibung eines „Fixpunkts“ von U, R, T
- … wobei wir weder U noch R noch T kennen!
- Beobachtete Aktions/Zustands/Reward-Sequenzen, wie eben

Wissen ausbauen oder ausbeuten?

… Englisch: *exploration vs. exploitation*

- „Mittendrin“ im Lernen
  haben wir approximative Nutzenwerte, Aktionsmodelle
- Sollen wir dann „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, finden 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 Nutzenwert
- n Häufigkeit, wie oft Zustand besucht wurde
- N feste Schranke
- R^+ feste Schätzung eines max. Rewards

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): $Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(a',s')$
- Q-Version der TD-Update-Regel

$$Q(a,s) \leftarrow Q(a,s) + \alpha\left(R(s) + \gamma \max_{a'} Q(a',s') - Q(a,s)\right)$$

(α kann von der Frequenz der Zustandsbesuche abhängen)

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

Vorausgesetzt geeignete Parameter der *f*-Funktion, approximiert die Funktion die Aktion eines optimalen MDP-Plans

Weiterführendes zum RL

- Will man modellfrei sein, oder will man eigentlich (auch) das Umgebungsmodell haben?
- Wie integriert man 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?

Gliederung

1. KI im Allgemeinen und in dieser Vorlesung
2. Heuristische Suche
3. Logik und Inferenz
4. Wissensrepräsentation
5. Handlungsplanung
6. Lernen
7. ~~Sprachverarbeitung~~
8. Umgebungswahrnehmung

Das war's!

Rückblick

**Die KI ist der Teil der Informatik,
der mittels algorithmischer Modelle Leistungen
des Denkens, Tuns und Wahrnehmens untersucht.**

Ausblick

- Bei der AG Wissensbasierte Systeme (Hertzberg)
  - Blockpraktikum „Mobile Robotik“, ab 14.2.2005
  - V+Ü *Wissensbasierte Robotik*, SS2005
  - Praktikum *RoboCup Rescue*, SS2005
  - Seminar Planungssysteme, SS2005
  - AG *Wissensbasierte Robotik*, Graduiertenseminar
- Veranstaltungen der AG Neuroinformatik (Riedmiller)
- Veranstaltungen im Studiengang Cognitive Science
- …