Lernen auf der Wissensebene als Beitrag zur Entwicklung Kognitiver Systeme

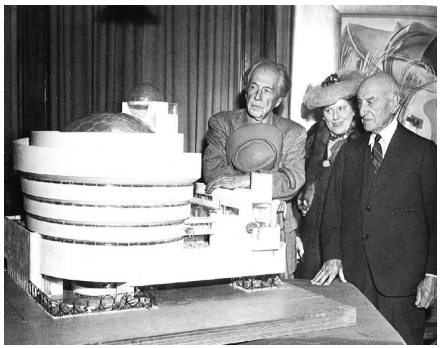
Ute Schmid

Kognitive Systeme
Fakultät Wirtschaftsinformatik und Angewandte Informatik
Otto-Friedrich Universität Bamberg

Gießen, 16.2.2011

Expertise

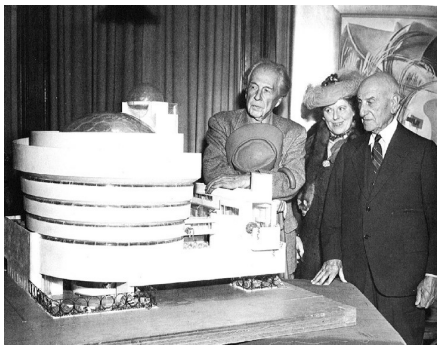
An expert is a man who has stopped thinking –
he knows! *(Frank Lloyd Wright)*



Frank Lloyd Wright, Hilla Rebay, Solomon Guggenheim at the unveiling of the model for the Guggenheim Museum, August, 1945
`www.westporthistory.org`

Expertise

An expert is a man who has stopped thinking –
he knows! *(Frank Lloyd Wright)*



Frank Lloyd Wright, Hilla Rebay, Solomon Guggenheim at the unveiling of the model for the Guggenheim Museum, August, 1945
`www.westporthistory.org`

- *gut feeling*: automatisierte, schnelle Klassifikation
- Schnelle, fehlerfreie, optimale Generierung von Problemlösungen: Anwendung komplexer Handlungsregeln
- Induktiver Erwerb von Wissen aus Erfahrung

Induktives Lernen

Aus Perspektive der Informatik: Maschinelles Lernen

- Ansatz zur Wissensakquisition (statt direktem *knowledge engineering*)
- Möglichkeit, um adaptive, flexible Systeme zu entwickeln
- Viele erfolgreiche Ansätze, insbesondere seit den 90er Jahren: Entscheidungsbaumalgorithmen (ID3, C4.5), künstliche neuronale Netze, SVM, Bayes-Klassifikatoren, kNN, Reinforcement Learning, genetische/evolutionäre Ansätze, Inductive Logic Programming, ...

Aus Perspektive der Kognitionswissenschaft

- *How can a cognitive system process environmental input and stored knowledge so as to benefit from experience?* (Holland, Holyoak, Nisbett & Thagard, 1986, Induction – Processes of Inference, Learning, and Discovery)

Induktives Lernen

Wechselseitige Befruchtung von KI und Psychologie

- Erste Entscheidungsbaum-Algorithmen von Hunt, Marin and Stone (1966) angeregt durch Studien zum Konzepterwerb von Bruner, Goodnow & Austin (A Study of Thinking, 1956)
- Parallele Entwicklungen: Exemplar-basierte Ansätze in der Psychologie (Nosofski), *lazy learning* (z.B. *k*NN) im maschinellen Lernen

- Lernen findet auf vielen Ebenen statt:
  Basale Lernprozesse (auf Sensordaten) —
  Symbolbasierte Ansätze (auf der Wissensebene)
- Fokus: Lernen auf der Wissensebene, speziell **produktive Regeln**

Produktive Regeln

- Klassifikation in potentiell unendlichen Domänen
  korrekt sortierte Liste, *korrekte Verkabelung*
- Erzeugung von Aktionsfolgen in verschieden komplexen Situationen
  Sortieren einer Liste, *Konstruktion eines Verkabelungsplans*

- Automatismen, die das Arbeitsgedächtnis kaum belasten
  (vgl. prozedurales/implizites Wissen)
- Aber: zum Teil kommunizierbar (symbolisch repräsentiert)
- In kognitiv technischem System: symbolische Regeln,
  Erklärungskomponente

Induktive Programmsynthese als Zugang zum Lernen produktiver Regeln

Induktive Programmierung

- Anwendung von maschinellem Lernen auf das Problem der Programmsynthese
- Automatische Erzeugung (rekursiver) Programme aus Eingabe-/Ausgabe-Beispielen (unvollständigen Spezifikationen)
- Ansätze:
  - analytisch, daten-getrieben: Thesys, Golem, Igor (ILP und induktive funktionale Ansätze)
  - generate-and-test: FFoil (ILP), Adate (evolutionär), MagicHaskeller (systematische Aufzählung)

E/A Beispiele:

last [a]	=	a
last [a,b]	=	b
last[a,b,c]	=	c
last[a,b,c,d]	=	d

Induziertes Programm:

last[x]	=	x
last(x:xs)	=	last(xs)

IGOR2

- Analytischer Ansatz zum Lernen funktionaler Programme (MAUDE, HASKELL)
- Effiziente Induktion rekursiver Regelmengen aus wenigen positiven Beispielen
- Lernt lineare, Baum-, und wechselseitig rekursive Regelmengen
- Automatische Induktion von Hilfsfunktionen (*necessary function invention*)
- Kann Hintergrundwissen berücksichtigen (analog zum ILP-System GOLEM)
- Restriction Bias: funktionale rekursive Programme, bei denen die äußere Funktion entweder nicht-rekursiv oder aus dem Hintergrundwissen ist
- Nicht-rekursive Programme zur Klassifikation als Spezialfall (z.B. `playTennis`, Mitchell, 1997)
- Präferenz-Bias: Weniger Fallunterscheidungen, spezifischere Patterns, weniger rekursive Aufrufe

(Kitzelmann and Schmid, JMLR 2006; Schmid et al., AGI 2009, Hofmann et al. KI 2008, Kitzelmann, AAIP 2009)

Empirische Ergebnisse

	isort	*reverse*	*weave*	*shiftr*	*mult/add*	*allodds*
ADATE	70.0	78.0	80.0	18.81	—	214.87
FLIP	×	—	$134.24^{\perp}$	$448.55^{\perp}$	×	×
FFOIL	×	—	$0.4^{\perp}$	$< 0.1^{\perp}$	$8.1^{\perp}$	$0.1^{\perp}$
GOLEM	0.714	—	$0.66^{\perp}$	0.298	—	$0.016^{\perp}$
IGOR II	0.105	0.103	0.200	0.127	⊙	⊙
MAGH.	0.01	0.08	⊙	157.32	—	×

	lasts	*last*	*member*	*odd/even*	*multlast*
ADATE	822.0	0.2	2.0	—	4.3
FLIP	×	0.020	17.868	0.130	$448.90^{\perp}$
FFOIL	$0.7^{\perp}$	0.1	$0.1^{\perp}$	$< 0.1^{\perp}$	< 0.1
GOLEM	1.062	< 0.001	0.033	—	< 0.001
IGOR II	5.695	0.007	0.152	0.019	0.023
MAGH.	19.43	0.01	⊙	—	0.30

— not tested × stack overflow ⊙ timeout $\perp$ wrong

all runtimes in seconds

IGOR2 als *Cognitive Rule Acquisition Device*

- Analytische induktive Programmierung bietet einen allgemeinen Mechanismus um generalisierte Regelmengen aus Beispielen für ein gewünschtes Verhalten zu extrahieren
- Typische Bereiche, in denen aus positiven Beispielen gelernt wird
  - Semantische Relationen (Ist-Ein, Vorfahr)
  - Sprache (regelmäßige Beugungen, grammatische Strukturen)
  - Problemlösen (Turm von Hanoi)
- Plötzliches Gefühl, ein Problem verstanden zu haben (Aha-Effekt): Regularitäten wurden zu einem (Problemlöse-) Schema generalisiert

Beispiel: Blockswelt

- **Ziel:** Baue einen Turm aus Blöcken in einer gegebenen Reihenfolge
- **Gegeben:** beliebiger Anfangszustand, beliebig viele Blöcke
- **Strategie:** *Räume denjenigen Block frei, der ganz unten liegen soll und stelle ihn auf den Tisch, stelle dann immer den nächst-benötigten Block auf den Teilturm und sorge dafür, dass dieser Block frei ist.*
- (Nicht-optimale Strategie: Lege erst alle Blöcke auf den Tisch und baue dann den Turm mit gewünschter Reihenfolge der Blöcke auf.)

Tower (9 examples of towers with up to four blocks, 1.2 sec)
(10 corresponding examples for Clear and IsTower as BK)

```
Tower(0, S) = S   if IsTower(0, S)
Tower(0, S) = put(0, Sub1(0, S),
              Clear(0, Clear(Sub1(0, S),
                     Tower(Sub1(0, S), S))))   if not(IsTower(0, S))

Sub1(s(0), S) = 0 .
```

Lernen aus Problemlöseerfahrung

- Generierung von optimalen Plänen für "kleine" Probleme, z.B. mit einem PDDL-Planer:
  - *Tower*: Türme mit bis zu vier Blöcken
  - *Turm von Hanoi*: ein bis drei Scheiben
  - *Rocket*: ein bis drei Kisten
- IGOR2 liefert generalisierte Problemlösestrategie
- Ergebnis: Problemlösen durch Suche wird unnötig

(Schmid, Wysotzki, AIPS 2000; Schmid et al. AGI 2009; Schmid & Kitzelmann, CSR 2011)

Vergleich mit anderen Ansätzen

- Generalisierung über vorhandene Problemlöseerfahrung statt suchbasiertes Erzeugen möglicher Strategien!

- Automatische Induktion generalisierter Handlungsregeln
- In kognitiven Architekturen (ACT-R, SOAR) stattdessen nur:
  - Veränderung von Stärkewerten
  - Chunking von Regeln zu komplexeren Einheiten
- Kein Erwerb neuer Regeln!

Induktiver Erwerb von Schlussregeln

Ancestor (9 examples, 10.1 sec)
(and corresponding 4 examples for IsIn and Or)

```
Ancestor(X, Y, nil) = nilp .
Ancestor(X, Y, node(Z, L, R)) =
  IsIn(Y, node(Z, L, R))  if X == Z .
Ancestor(X, Y, node(Z, L, R)) =
  Ancestor(X, Y, L) Or Ancestor(X, Y, R)  if X =/= Z .
```

Corresponding to:
ancestor(x,y) = parent(x,y).
ancestor(x,y) = parent(x,z), ancestor(z,y).
isa(x,y) = directlink(x,y).
isa(x,y) = directlink(x,z), isa(z,y).

Erwerb der Schlussregel für Transitivität (*modus barbara*)

Grammatiklernen

examples

```
fmod GENERATOR is
  *** types
  sorts Cat CList Depth .
  ops d n v : -> Cat [ctor] .
  op ! : -> CList [ctor] .
  op __ : Cat CList -> CList [ctor] .
  op 1 : -> Depth [ctor] .
  op s_ : Depth -> Depth [ctor] .
  *** target fun declaration
  op Sentence : Depth -> CList [metadata "induce"] .
  *** examples
  eq Sentence(1) = (d n v d n !) .
  eq Sentence(s 1) = (d n v d n v d n !) .
  eq Sentence(s s 1) = (d n v d n v d n v d n !) .
```

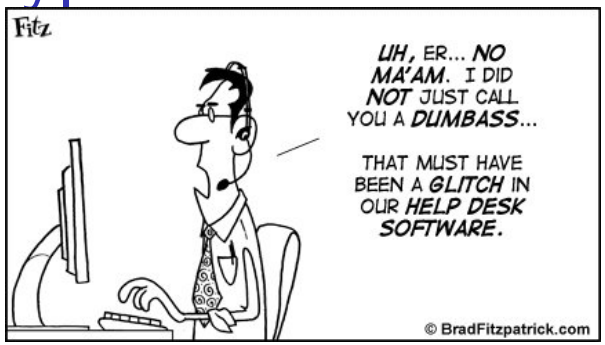
learned grammar rules (3 examples, 0.072 sec)

```
Sentence(1) = (d n v d n !)
Sentence(s N) = (d n v Sentence(N))
```

corresponds to simple phrase structure grammar

```
S -> NP VP
NP -> d n
VP -> v NP | v S
```

Lernen Struktureller Prototypen

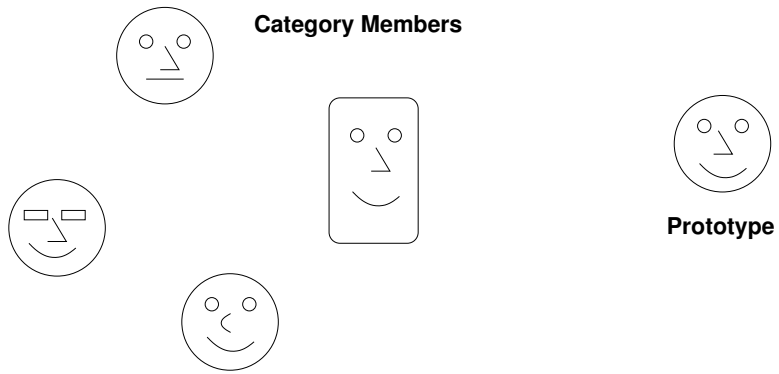


- Zusammenarbeit mit SAP: Incident-Mining als Vorbereitung auf Business ByDesign
- Support ist häufig über viele Personen und Orte verteilt
- hoher Kostenfaktor, kann ausschlaggebend für Kundenzufriedenheit sein
- Problem: Kenntnis der Support-Ingenieure über bereits vorhandene Lösungen
- Intelligente Software-Unterstützung für Support-Ingenieure:
  - Automatischer Support bei Standard-Problemen
  - Zugang zu existierenden Lösungen für bereits aufgetretene und behobene Probleme

(Schmid et al., IEA-AIE 2010)

Prototyp-Lernen

- Cluster-Verfahren helfen, Gruppen ähnlicher Objekte zu identifizieren (unüberwachtes Lernen)
- Clustering ist ein Ansatz zum Konzepterwerb
- Explizite Konstruktion symbolischer Prototypen für jedes Cluster
  - kann Klassifikationszeit reduzieren
  - erlaubt Experten die Gemeinsamkeiten von Objekten in einem Cluster zu inspizieren
  - ist angeregt durch psychologische Befunde zum menschlichen Konzepterwerb (Roschs Prototypen-Theorie)



Structural Incident Mining

- Typische Ansätze im *data mining* sind merkmalsbasiert
- *Incident reports liegen als XML-Bäume* – basierend auf einem Incident Modell – vor
  $\hookrightarrow$ Transformation von Strukturen in Merkmale führen immer zu einem Informationsverlust
- Unser Ansatz: Extraktion *struktureller* Prototypen
  - Anti-unification (AU): Schnittmenge von Bäumen
  - Structure dominance generalisation (SDTG): Vereinigung von Bäumen
  - SDTG-AU: moderate Generalisierung; Generalisierung über Knoten mit verschiedenen Labeln, Berücksichtigung von Auftretenshäufigkeiten von Labeln

Ergebnisse

Leave-one-out

Method	Hits	Errors	Av. Size	Generation (sec.)	Retrieval (sec.)
FOIL	11	8, 14[1]		0,109	0,409
AU	29	4	173	0,485	0,044
SDTG	22	11	387	0,456	0,049
SDTG-AU	31	2	264	0,419	0,047

Noisy Data

Method	Hits	Errors	Av. Size	Generation (sec.)	Retrieval (sec.)
FOIL	0	0, 57[1]		0,020	7,378
AU	4	53	66	0,202	0,051
SDTG	52	5	619	0,227	0,102
SDTG-AU	56	1	410	0,188	0,069

[1] first value: classification error, second value: ambiguous result

(Retrieval for AU based on subsumption, retrieval for SDTG/SDTG-AU on Manhatten distance)

Schmerzklassifikation

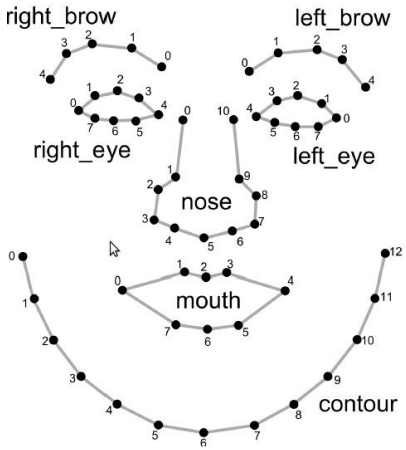
Klassifikation von Schmerz aus Mimikdaten

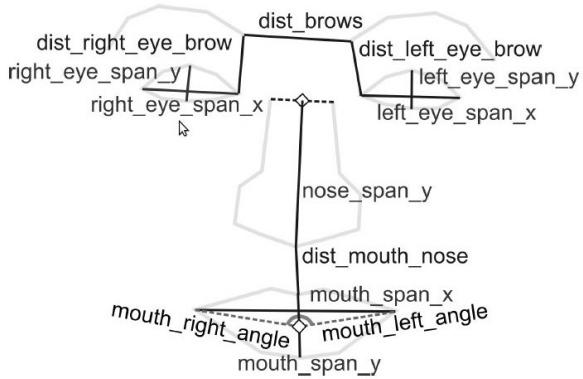
- Zusammenarbeit mit der Biopsychologie (Lautenbacher & Kunz)
- Empirische Belege, dass Mimik ein guter Prädiktor für Schmerz ist (in Abwesenheit der Möglichkeit von Selbstauskünften)
- Anwendungsszenarien: Post-operativ, Personen mit dementiellen Syndromen (Altzheimer)
- Typisches Vorgehen: Identifikation von *action units* (Facial Action Coding System, FACS, Ekman & Friesen)
- Bei der Entwicklung von FACS wurde Schmerz nicht berücksichtigt

Schmerzklassifikation

Bisherige Arbeiten: Identifikation relevanter Merkmale

- Betrachtung einer großen Menge von Distanz- und Winkel-Relationen zwischen relevanten Punkten im Gesicht
- Lernen von Klassifikatoren über Bilder, die in psychologischen Experimenten mit Induktion von Druckschmerz erhoben wurden
- Zentrale Fragen:
  - Vergleich von gemittelten und individuellen Klassifikatoren
  - Vergleich der Merkmalsmengen mit FACS





(Siebers, Kunz, Lautenbacher, Schmid, KI-WS 2009; KDIR 2010)

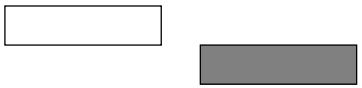
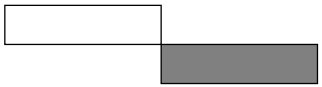
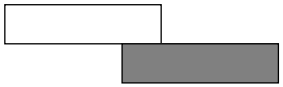
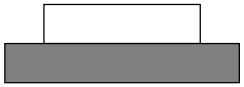
Sind zeitliche Abfolgen relevant?

- Steckt diagnostisch relevante Information in den zeitlichen Beziehungen des Auftretens von *action units* (AUs)?
- Konventionelles Vorgehen beim *temporal data mining*: Extraktion von Mustern aus Sequenzen
  Informationsverlust: Nur Abfolge, nicht Beziehungen zwischen Ereignissen werden berücksichtigt
- Vorschlag
  - Repräsentation der Relationen zwischen AUs unter Verwendung des Allen Kalküls
  - Nutzung von ILP Verfahren, um relevante zeitlichen Relationen zu identifizieren

Allen Kalkül

- James F. Allen, 1983, Maintaining Knowledge about temporal intervals, *Communications of the ACM*, *26*(*1*), 832-843.
- $1D$, intervall-basierter Kalkül
- Jedes Intervall X ist ein geordnetes Paar von Startpunkten s_X und Endpunkten e_X
- Interpretation: Abbildung auf reele Zahlen mit $s_X < e_X$
- 13 Basisrelationen: paarweise disjunkt und exhaustiv
- Komposition von Relationen definiert

Allen Calculus

Symbol	Description	Visualization	Point Order
$X \prec Y$	X before Y		$s_X < e_X < s_Y < e_Y$
$X\ m\ Y$	X meets Y		$s_X < e_X = s_Y < e_Y$
$X\ o\ Y$	X overlaps Y		$s_X < s_Y < e_X < e_Y$
$X\ s\ Y$	X starts Y		$s_Y = s_X < e_X < e_Y$
$X\ d\ Y$	X during Y		$s_Y < s_X < e_X < e_Y$
$X\ f\ Y$	X finishes		$s_Y < s_X < e_X = e_Y$
$X = Y$	X equals Y		$s_X = s_Y < e_Y = e_X$

and inverse relations (fi, di, si, oi, mi, $\succ$)

Beispiel

Rohdaten

Tick	ON	OFF
17	4	-
18	-	-
19	5	-
20	-	4
21	-	-

Repräsentation

`during(5,4)`

- Transformation der Rohdaten in Allen-Relationen über die Start- und Endzeiten (plus Zeitfenster)

Trainingsdaten

```
pain(id1). % positive example
occurs(id1,au3). % AUs
occurs(id1,au4).
occurs(id1,au5).
occurs(id1,au7).
before(id1, au3, au4). % relations
during(id1, au5, au4).
after(id1, au4, au7).
```

Induzierte Regeln

pain(Id) :- occurs(Id, au3), occurs(Id, au4),
occurs(Id, au5), during(au5, au4).

pain(Id) :- occurs(Id, au1), occurs(Id, au4),
before(Id, au4, au5).

Temporal Pattern Mining

Nutzung eines logik-basierten Ansatzes

- Erlaubt reichhaltigere Information beim Lernen von Klassifikatoren zu verwenden
- Ergebnisse könnten von Interesse für die Schmerzforschung sein: Helfen zeitliche Beziehungen zwischen AUs Schmerz von anderen Ausdrücken zu unterscheiden? („Grammatik“)
- Symbolisches Lernverfahren: Gelernte Regeln können von menschlichen Experten inspiziert werden
- Gelernte Regeln können auf natürliche Weise in wissensbasierte Systeme integriert werden

Zusammenfassung

- Induktives Lernen und induktiver Wissenserwerb
  - Lernen rekursiver Regelmengen: Erwerb von Problemlösestrategien
  - Lernen stuktureller Prototypen als anschauliche Cluster-Repräsentanten
  - Identifikation relevanter zeitlicher Beziehungen am Beispiel von Schmerzmimik
- Symbolische Lernverfahren adressieren Lernen auf der Wissensebene
- Beitrag zur Kognitiven Modellierung (in typischen kognitiven Architekturen wird Lernen nur rudimentär behandelt)
- Beitrag zur Wissensaquisition für wissensbasierte Systeme: statt *black box* Klassifikatoren inspizierbare Regeln; nutzbar zur Erklärung von Systementscheidungen