

Buch zu den Folien hilft für Wiederholung

Folien zum Buch

Grundkurs Künstliche Intelligenz

Wolfgang Ertel

Vieweg-Verlag, November 2007

www.hs-weingarten.de/~ertel/kibuch

www.vieweg.de

Angenommen A ist Aussagenvariable
Virtuell:
 $A \quad | \quad \text{nicht } A \wedge A$
 $\top \quad | \quad \underline{\quad}$

Mittelwert, Standardabweichung
Grunds - Ausrisse - Test bestreben (Steps / Konzept)

13. 02. 11:00 Uhr

Inhaltsverzeichnis

Literaturverzeichnis	höhe Präzision in Logikbereich (wie ist Formel definiert)	3
1 Einführung		12
2 Aussagenlogik x	2.	35
3 Prädikatenlogik erster Stufe x	3.	71
4 Grenzen der Logik	4. Machine Learning	137
5 Logikprogrammierung mit PROLOG		158
6 Suchen, Spielen und Probleme lösen		187
7 Schließen mit Unsicherheit		242
8 Maschinelles Lernen und Data Mining		342
9 Neuronale Netze		484
10 Lernen durch Verstärkung (Reinforcement Learning)		563

Literaturverzeichnis

The RoboCup Soccer Simulator. <http://sserver.sourceforge.net> 3

Alpaydin, E.: Introduction to Machine Learning. MIT Press, 2004 8.8, 9.6

Anderson, J./Pellionisz, A./Rosenfeld, E.: Neurocomputing (vol. 2): directions for research. Cambridge, MA, USA: MIT Press, 1990 9.8

Anderson, J./Rosenfeld, E.: Neurocomputing: Foundations of Research. Cambridge, MA: MIT Press, 1988, Sammlung von Originalarbeiten ([document](#)), 1, 9.8

Bartak, R.: Online Guide to Constraint Programming. <http://kti.ms.mff.cuni.cz/\protect\unhbox\voidb@x\penalty\@M\{}bartak/constraints>, 1998 5.8

Barto, A. G./Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discrete Event Systems, Special issue on reinforcement learning, 13 2003, 41–77 **10.1**

Bellman, R.E.: Dynamic Programming. Princeton University Press, 1957 **10**

Berrondo, M.: Fallgruben für Kopffüßler. Fischer Taschenbuch Nr. 8703, 1989 **2.4**

Bibel, W.: Deduktion: Automatisierung der Logik. Band 6.2, Handbuch der Informatik. Oldenbourg, 1992 **3.5, 3.5, 3.9**

Billard, A. et al.: Robot Programming by Demonstration. In **Siciliano, B./Khatib, O. (Hrsg.)**: Handbook of Robotics. Springer, 2008, 1371–1394 **10.2**

Bläsius, K.H./Bürckert, H.-J.: Deduktionssysteme. Oldenbourg, 1992 **3.5, 3.9**

Bratko, I.: PROLOG: Programmierung für Künstliche Intelligenz. Addison-Wesley, 1986 **5, 5.8**

Burges, C. J.: A Tutorial on Support Vector Machines for Pattern Recognition. Data Min. Knowl. Discov. 2 1998, Nr. 2, 121–167 **9.6**

Chang, C. L./Lee, R. C.: Symbolic Logic and Mechanical Theorem Proving.
Orlando, Florida: Academic Press, 1973 **3.9**

Clocksin, W. F./Mellish, C. S.: Programming in Prolog. 4. Auflage. Berlin,
Heidelberg, New York: Springer, 1994 **5, 5.8**

Dassow, J.: Logik für Informatiker. Teubner Verlag, 2005 **3.9**

Diaz, D.: GNU PROLOG. Universität Paris, 2004, Aufl. 1.7, für GNU Prolog
version 1.2.18, <http://gnu-prolog.inria.fr> **5.1, 5.8**

D.J. Newman, S. Hettich, C.L. Blake/Merz, C.J.: UCI Repository of ma-
chine learning databases. <http://www.ics.uci.edu/\protect\unhbox\voidb@x\penalty\@M\{\}mlearn/MLRepository.html>, 1998 **8.8**

Duda, R.O./Hart, P.E.: Pattern Classification and Scene Analysis. Wiley, 1973,
Klassiker zur Bayes-Decision-Theorie ([document](#))

Duda, R.O./Hart, P.E./Stork, D.G.: Pattern Classification. Wiley, 2001, Neu-
auflage des Klassikers [Duda/Hart](#) **7.6, 16, 8.6, 19, 8.8**

Eder, E.: Relative Complexities of First Order Calculi. Vieweg Verlag, 1991 **3.3**

Ertel, W./Schumann, J./Suttnner, Ch.: Learning Heuristics for a Theorem Prover using Back Propagation. In **Retti, J./Leidlmaier, K. (Hrsg.)**: 5. Österreichische Artificial-Intelligence-Tagung. Berlin, Heidelberg: Informatik-Fachberichte 208, Springer-Verlag, 1989, 87–95 **4.1, 4, 8**

Fischer, B./Schumann, J.: SETHEO Goes Software Engineering: Application of ATP to Software Reuse. In Conference on Automated Deduction (CADE 97). Springer, 1997, <http://ase.arc.nasa.gov/people/schumann/publications/papers/cade97-reuse.html>, 65–68 **3.8**

Freuder, E.: In Pursuit of the Holy Grail. Constraints, 2 1997, Nr. 1, 57–61 **5.7**

Görz, G./Rollinger, C.-R./Schneeberger, J. (Hrsg.): Handbuch der Künstlichen Intelligenz. Oldenbourg Verlag, 2003 ([document](#)), **3.9, 8**

Jensen, F.V.: Bayesian networks and decision graphs. Springer-Verlag, 2001 **7.5, 7.5, 7.6, 8.4, 12**

Kaelbling, L.P./Littman, M.L./Moore, A.P.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, 4 1996, 237–285, www-2.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a.pdf **10.2**

Kalman, J.A.: Automated Reasoning with OTTER. Rinton Press, 2001,
www-unix.mcs.anl.gov/AR/otter/index.html 3.6

Lauer, M./Riedmiller, M.: Generalisation in Reinforcement Learning
and the Use of Observation-Based Learning. In **Kokai, Gabriella/Zeidler, Jens (Hrsg.)**: Proceedings of the FGML Workshop 2002. 2002,
<http://amy.informatik.uos.de/riedmiller/publications/lauer.riedml.fgml02.ps.gz>, 100–107 2

Letz, R. et al.: SETHEO: A High-Performance Theorem Prover. Journal of Automated Reasoning, 1992, Nr. 8, 183–212, www4.informatik.tu-muenchen.de/\protect\unhbox\voidb@x\penalty\@M\{}letz/setheo 3.6

Melancon, G./Dutour, I./Bousque-Melou, G.: Random Generation of Dags for Graph Drawing. Dutch Research Center for Mathematical and Computer Science (CWI), 2000 (INS-R0005). – Technischer Bericht, <http://ftp.cwi.nl/CWIreports/INS/INS-R0005.pdf> 13

Minsky, M./Papert, S.: Perceptrons. MIT Press, Cambridge, MA, 1969 3

Mitchell, T.: Machine Learning. McGraw Hill, 1997, www-2.cs.cmu.edu/\protect\unhbox\voidb@x\penalty\@M\{}tom/mlbook.html 2, 11,

8.8, 10.2

Nipkow, T./Paulson, L.C./Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic. Band 2283, LNCS. Springer, 2002, www.cl.cam.ac.uk/Research/HVG/Isabelle 3.6, 4.1

Palm, G.: On Associative Memory. *Biological Cybernetics*, 36 1980, 19–31 2

Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference. Morgan Kaufmann, 1988 7.5, 7.5, 7.6

Rich, E.: Artificial Intelligence. McGraw-Hill, 1983 1.1, 1

Richter, M.: Fallbasiertes Schließen. In **Görz/Rollinger/Schneeberger**: Handbuch der Künstlichen Intelligenz, 407–430 8

Ritter, H./Martinez, T./Schulten, K.: Neuronale Netze. Addison Wesley, 1991 9.8

Rojas, R.: Theorie der neuronalen Netze. Springer, 1993 9.8

Rumelhart, D./McClelland, J.: Parallel Distributed Processing. Band 1, MIT Press, 1986 ([document](#)), 3, 4, 5

Rumelhart, D.E./Hinton, G.E./R.J., Williams: Learning Internal Representations by Error Propagation. in [Rumelhart/McClelland, 1986](#) 3, 5

Russell, S./Norvig, P.: Artificial Intelligence: A Modern Approach. 2. Auflage. Prentice Hall, 2003, 1. Auflage 1995, deutsche Übersetzung der 2. Auflage 2004 bei Pearson Studium, <http://aima.cs.berkeley.edu> 7.5

Schramm, M.: Indifferenz, Unabhängigkeit und maximale Entropie: Eine wahrscheinlichkeitstheoretische Semantik für Nicht-Monotones Schließen. München: CS-Press, 1996, Dissertationen zur Informatik 4 7.5

Schulz, S.: E – A Brainiac Theorem Prover. Journal of AI Communications, 15 2002, Nr. 2/3, 111–126, www4.informatik.tu-muenchen.de/\protect\unhbox\voidb@x\penalty\@M\{\}schulz/WORK/eprover.html 3.6, 3.7

Schumann, J.: Automated Theorem Proving in Software Engineering. Springer Verlag, 2001 3.8

Schölkopf, S./Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, 2002 9.6

Sejnowski, T.J./Rosenberg, C.R.: NETtalk: a parallel network that learns to read aloud. The John Hopkins University Electrical Engineering and Computer

Science Technical Report, 1986 (JHU/EECS-86/01). – Technischer Bericht,
Wiederabdruck in Anderson/Rosenfeld S. 661-672 7

Siekmann, J./Benzmüller, Ch.: Omega: Computer Supported Mathematics.
In KI 2004: Advances in Artificial Intelligence. Springer Verlag, 2004,
LNAI 3238, www.ags.uni-sb.de/\protect\unhbox\voidb@x\penalty\@M\{}omega, 3–28 4.1

Stone, P./Sutton, R.S./Kuhlmann, G.: Reinforcement Learning for
RoboCup-Soccer Keepaway. Adaptive Behavior, 2005, To appear.,
www.cs.utexas.edu/\protect\unhbox\voidb@x\penalty\@M\{}pstone/Papers/bib2html-links/AB05.pdf 3

Suttner, Ch./Ertel, W.: Automatic Acquisition of Search Guiding Heuristics. In
10th Int. Conf. on Automated Deduction. Springer-Verlag, LNAI 449, 1990,
470–484 4.1, 4, 9

Sutton, R./Barto, A.: Reinforcement Learning. MIT Press, 1998,
www.cs.ualberta.ca/\protect\unhbox\voidb@x\penalty\@M\{}sutton/book/the-book.html 10.2, 1, 10.2

Turing, A.M.: Computing Machinery and Intelligence. Mind, 59 1950, 433–460,
Deutsche Übersetzung mit dem Titel **Kann eine Maschine denken** in Zimmerli/Wolf 14

Whittaker, J.: Graphical models in applied multivariate statistics. Wiley, 1996 7.6

Wiedemann, U.: PhilLex, Lexikon der Philosophie. www.phillex.de/paradoxa.htm 2

Wielemaker, J.: SWI-Prolog 5.4. Universität Amsterdam, 2004, www.swi-prolog.org 5.8

Witten, I./Frank, E.: Data Mining. Hanser Verlag München, 2001, Von den Autoren in Java entwickelte DataMining Programmbibliothek WEKA:
(www.cs.waikato.ac.nz/\protect\unhbox\voidb@x\penalty\@M\{}/weka) 8, 20

Zdziarski, J.: Ending Spam. No Starch Press, 2005 15

Zell, A.: Simulation Neuronaler Netze. Addison Wesley, 1994, Im Buch beschriebener Simulator SNNS, bzw. JNNS: www-ra.informatik.uni-tuebingen.de/SNNS 6, 9.8

Zimmerli, W.C./Wolf, S. (Hrsg.): Künstliche Intelligenz – Philosophische Probleme. Stuttgart: Philipp Reclam, 1994 ([document](#))

Kapitel 1

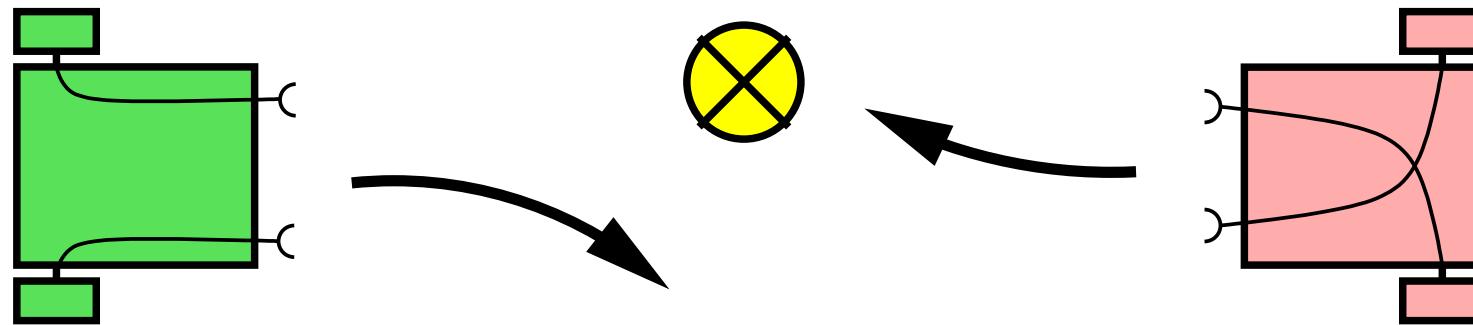
Einführung

Was ist Künstliche Intelligenz (KI)

- Was ist Intelligenz?
- Wie kann man Intelligenz messen?
- Wie funktioniert unser Gehirn?
- Intelligente Maschine?
- Science Fiction?
- Menschlichen Geist nachbauen?
- Philosophie? z.B. Körper–Seele Problem

John McCarthy (1955):

Ziel der KI ist es, Maschinen zu entwickeln, die sich verhalten, als verfügten sie über Intelligenz.



Zwei ganz einfache Braitenberg-Vehikel und deren Reaktion auf eine Lichtquelle.

Encyclopedia Britannica:

KI ist die Fähigkeit digitaler Computer oder computergesteuerter Roboter, Aufgaben zu lösen, die normalerweise mit den höheren intellektuellen Verarbeitungsfähigkeiten von Menschen in Verbindung gebracht werden ...

Nach dieser Definition ist also jeder Computer ein KI-System.

Elaine Rich Rich:

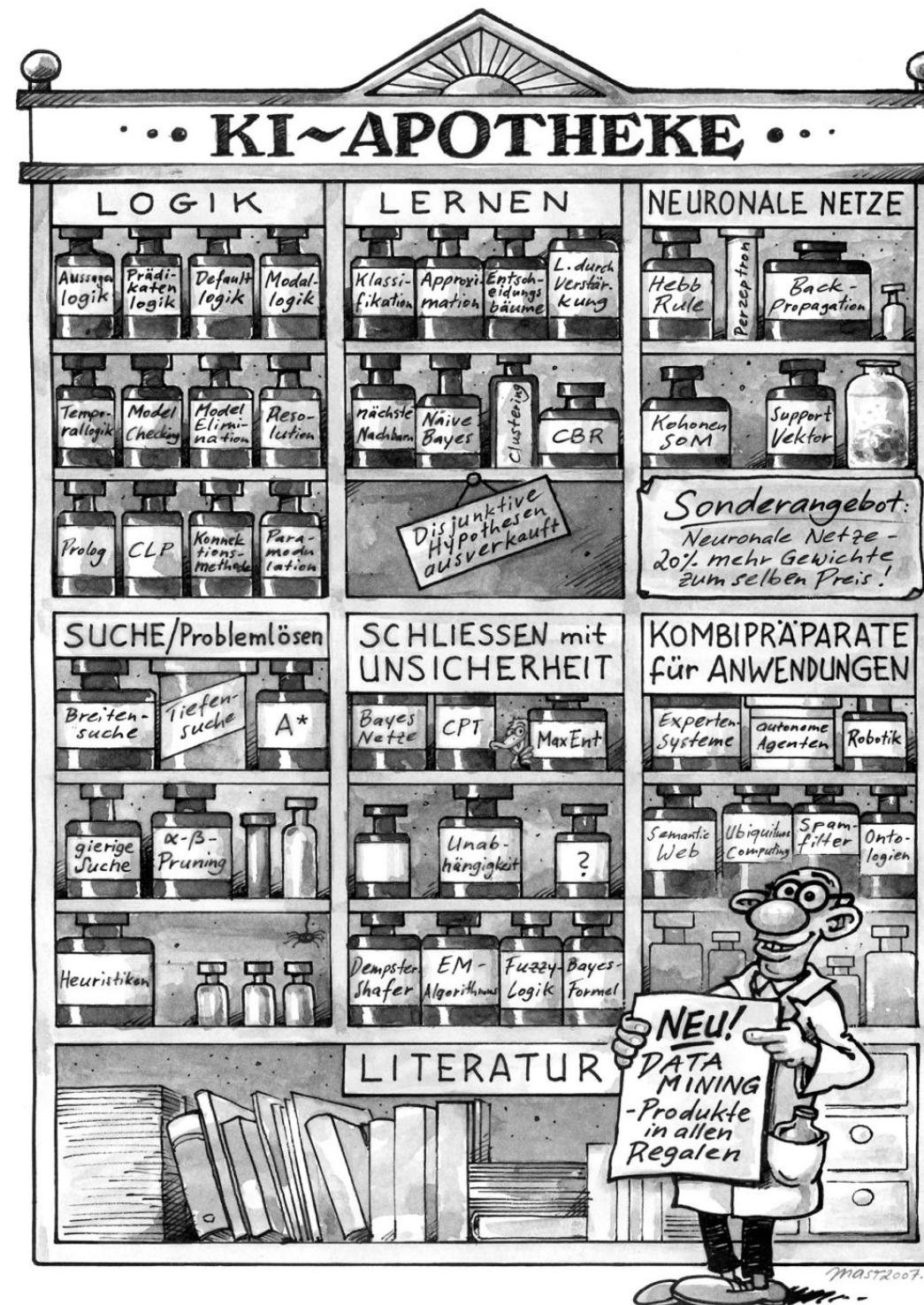
Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.

- Auch im Jahr 2050 noch aktuell!
- Mensch heute noch besser in vielen Bereichen!
u.a. Bildverstehen, Lernfähigkeit
- Computer heute schon besser in vielen Bereichen!
- Beisp. Schachcomputer

Hirnforschung und Problemlösen

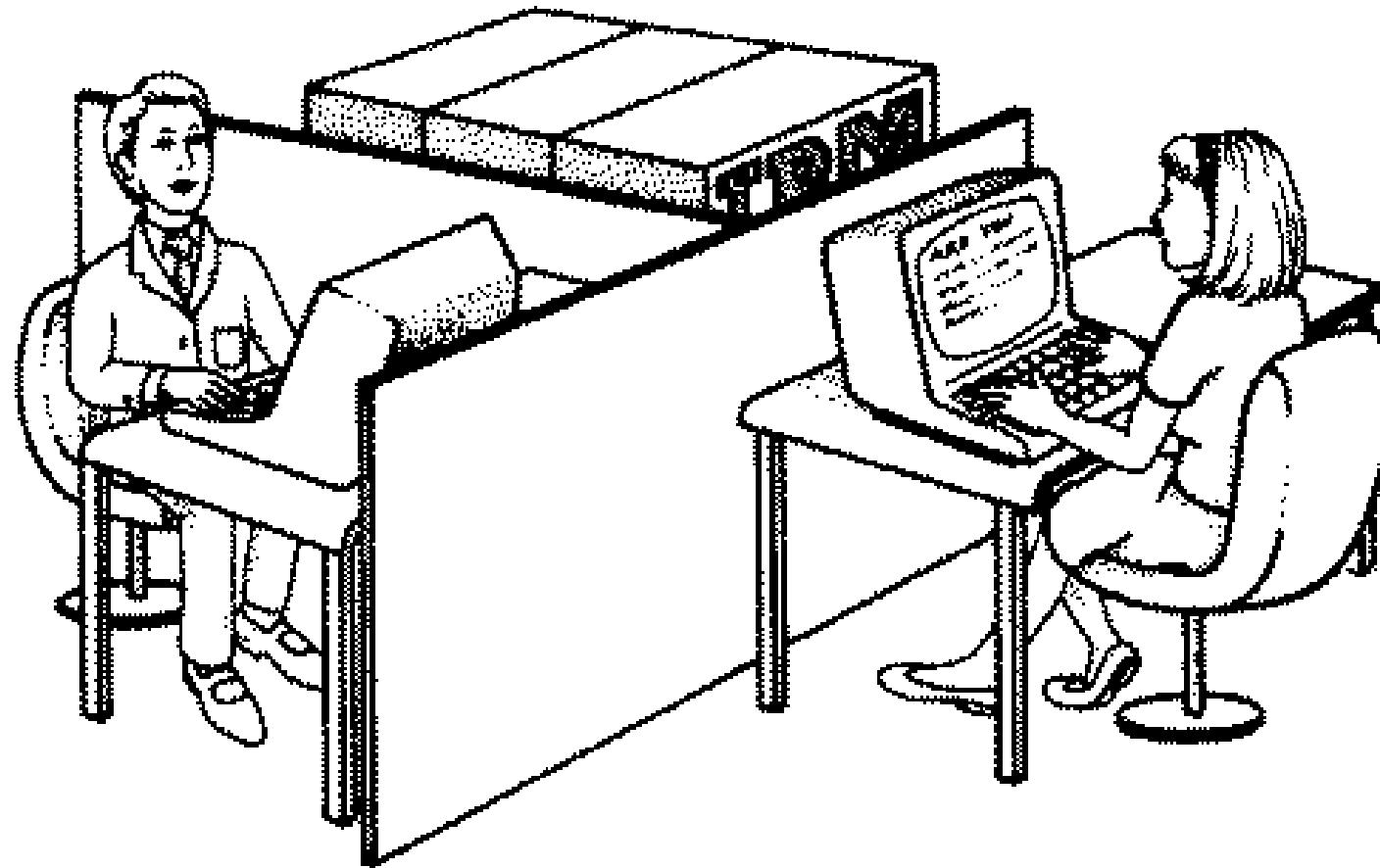
unterschiedliche Ansätze:

- Wie arbeitet das menschliche Gehirn?
- Problemorientiert: Intelligente Agenten bauen!
- Gemischtwarenladen!



Der Turingtest und Chatterbots

Alan Turing:



Die Maschine besteht den Test, wenn sie Alice in 30% der Fälle täuschen kann.

Joseph Weizenbaum (Computerkritiker): Programm Eliza spricht mit seiner Sekretärin

[Demo: Chatterbot](#)

[Demo: Noch ein Chatterbot](#)

Geschichte der KI

- 1931** Der Österreicher Kurt Gödel zeigt, dass in der **Prädikatenlogik** erster Stufe alle wahren Aussagen herleitbar sind. In Logiken höherer Stufe hingegen gibt es wahre Aussagen, die nicht beweisbar sind .
- 1937** Alan Turing zeigt mit dem Halteproblem Grenzen intelligenter Maschinen auf .
- 1943** McCulloch und Pitts modellieren **Neuronale Netze** und stellen die Verbindung zur Aussagenlogik her.
- 1950** Alan Turing definiert über den **Turingtest** Intelligenz von Maschinen und schreibt über lernende Maschinen und genetische Algorithmen
- 1951** Marvin Minsky entwickelt einen Neuronenrechner. Mit 3000 Röhren simuliert er 40 Neuronen.
- 1955** Arthur Samuel (IBM) baut lernfähige Dame Programme die besser spielen als ihre Entwickler

1956 McCarthy organisiert eine Konferenz im Dartmouth College. Hier wird der Name **Artificial Intelligence** festgelegt.

Newell und Simon von der Carnegie Mellon University (CMU) stellen den **Logic Theorist**, das erste symbolverarbeitende Programm, vor .

1958 McCarthy erfindet am MIT (Massachusetts Institute of Technology) die Hochsprache **LISP**. Er schreibt Programme, die sich selbst verändern können.

1959 Gelernter (IBM) baut den Geometry Theorem Prover.

1961 Der General Problem Solver (GPS) von Newell und Simon imitiert menschliches Denken .

1963 McCarthy gründet das AI-Lab an der Stanford Universität.

1965 Robinson erfindet den **Resolutionkalkül** für Prädikatenlogik (??).

1966 Weizenbaum's Eliza-Programm führt Dialoge mit Menschen in natürlicher Sprache (??).

1969 Minsky und Papert zeigen in ihrem Buch **Perceptrons** auf, dass das Perzepron, ein sehr einfaches neuronales Netz, nur lineare Zusammenhänge repräsentieren kann (1.1).

1972 Der Franzose Alain Colmérauer erfindet die Logikprogrammiersprache **PROLOG** (5).

Der britische Mediziner de Dombal entwickelt ein **Expertensystem** zur Diagnose von Bauchkrankheiten . Es blieb in der bis dahin überwiegend amerikanischen KI-Community unbeachtet (??).

1976 Shortliffe und Buchanan entwickeln MYCIN, ein Expertensystem zur Diagnose von Infektionskrankheiten, das mit Unsicherheit umgehen kann (??).

1981 Japan startet mit großem Aufwand das “Fifth Generation Project” mit dem Ziel, leistungsfähige intelligente PROLOG-Maschinen zu bauen.

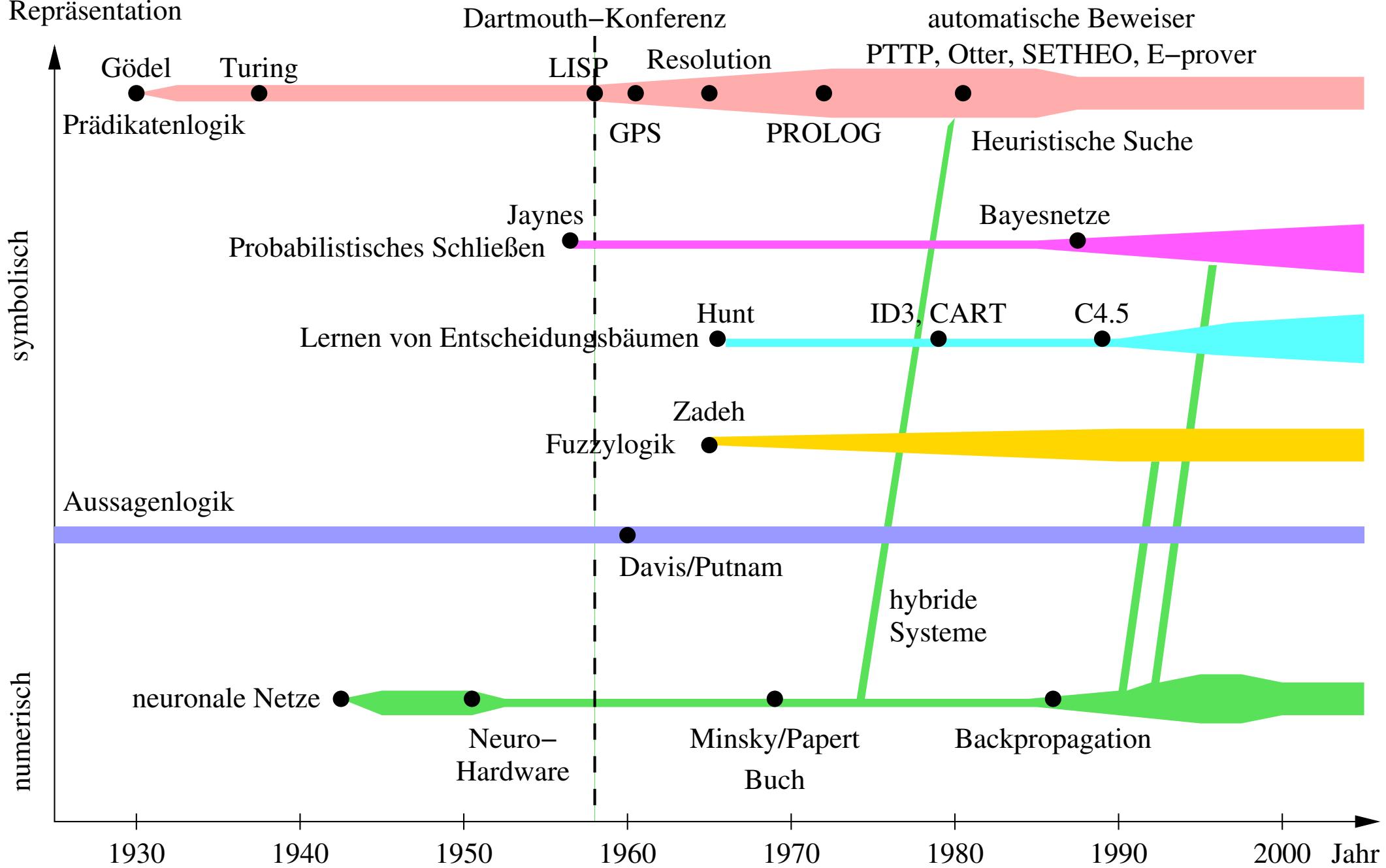
1982 Das Expertensystem R1 zur Konfiguration von Computern spart der Digital Equipment Corporation 40 Millionen Dollar pro Jahr .

- 1986** Renaissance der Neuronalen Netze unter anderem durch Rumelhart, Hinton und Sejnowski . Das System Nettalk lernt das Vorlesen von Texten. (9).
- 1990** Pearl , Cheeseman , Whittaker, Spiegelhalter bringen mit den **Bayes-Netzen** die Wahrscheinlichkeitstheorie in die KI (7.5).
Multiagentensysteme werden populär.
- 1993** Weltweite **RoboCup** Initiative zum Bau Fußball spielender autonomer Roboter (??).
- 1997** Erster internationaler RoboCup Wettkampf in Japan.
- 2003** Die Roboter im RoboCup demonstrieren eindrucksvoll, was KI und Robotik zu leisten imstande sind.

Die Phasen der KI-Geschichte

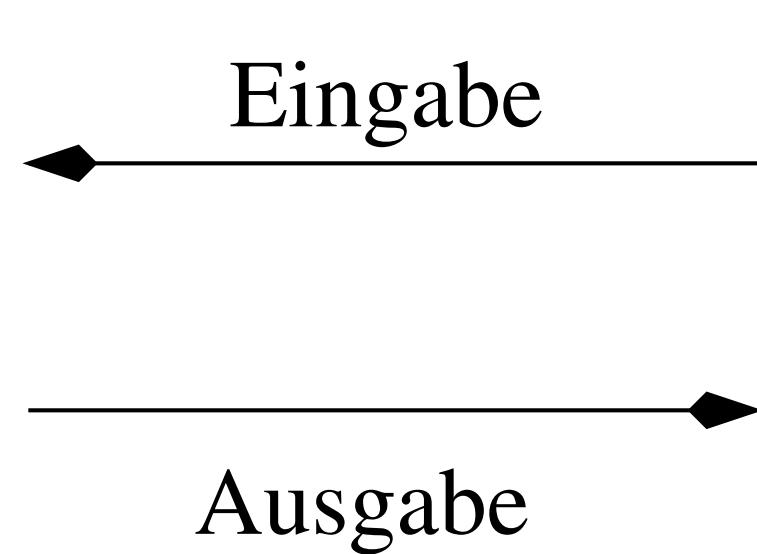
- Die ersten Anfänge
- Logik löst (fast) alle Probleme
- Der neue Konnektionismus
- Schließen mit Unsicherheit
- Verteilte, autonome und lernende Agenten
- Die KI wird erwachsen

Mächtigkeit der Repräsentation



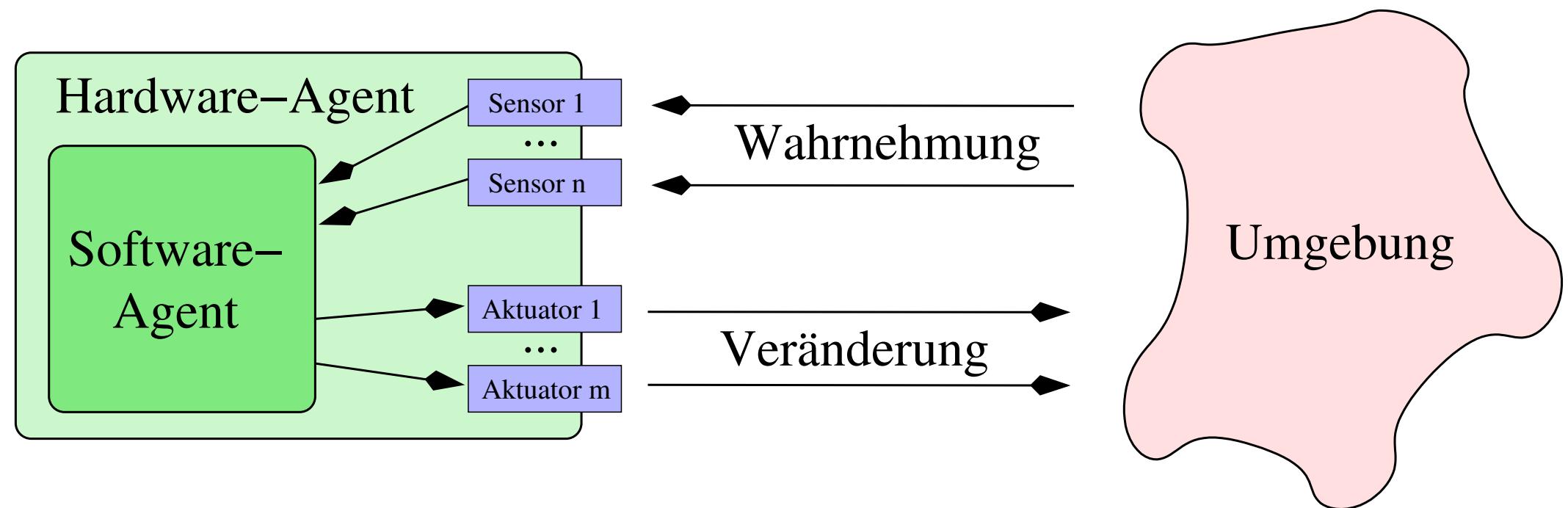
Agenten

Software-Agent



Benutzer

Hardware-Agent (autonomer Roboter)



Reflex-Agent: Funktion von der Menge aller Eingaben auf die Menge aller Ausgaben

Agent mit Gedächtnis: ist keine Funktion. Warum?

lernfähiger Agent

verteilte Agenten

Markov-Entscheidungsprozess: zur Bestimmung der optimalen Aktion wird nur der aktuelle Zustand benötigt.

zielorientierter Agent

Beispiel: **Spamfilter:** Sein Ziel ist es, Emails in die richtige Klasse einzuteilen.

Agent 1:

		korrekte Klasse	
		erwünscht	SPAM
Spamfilter entscheidet	erwünscht	189	1
	SPAM	11	799

Agent 2:

		korrekte Klasse	
		erwünscht	SPAM
Spamfilter entscheidet	erwünscht	200	38
	SPAM	0	762

Ist Agent 2 schlechter als Agent 1?

Definition 1.2 Ziel eines **kostenorientierten Agenten** ist es, die durch Fehlentscheidungen entstehenden Kosten langfristig, das heißt im Mittel, zu minimieren. Die Summe aller gewichteten Fehler ergibt die durch die Fehlentscheidungen entstandenen Kosten.

Beispiel: Blinddarmdiagnosesystem LEXMED Abschnitt ??.

Analog ist das Ziel eines **nutzenorientierten Agenten** (engl. utility based agent), den durch korrekte Entscheidungen entstehenden Nutzen langfristig, das heißt im Mittel, zu maximieren.

Umgebung

- beobachtbar (Schachcomputer)
- teilweise beobachtbar (Roboter)
- deterministisch (8-Puzzle)
- nichtdeterministisch (Schachcomputer, Roboter)
- diskret (Schachcomputer)
- stetig

Wissensbasierte Systeme

strenge Trennung:

Wissen

Inferenzmechanismus

Wissensbasis, kurz *WB* (engl. knowledge base, KB)

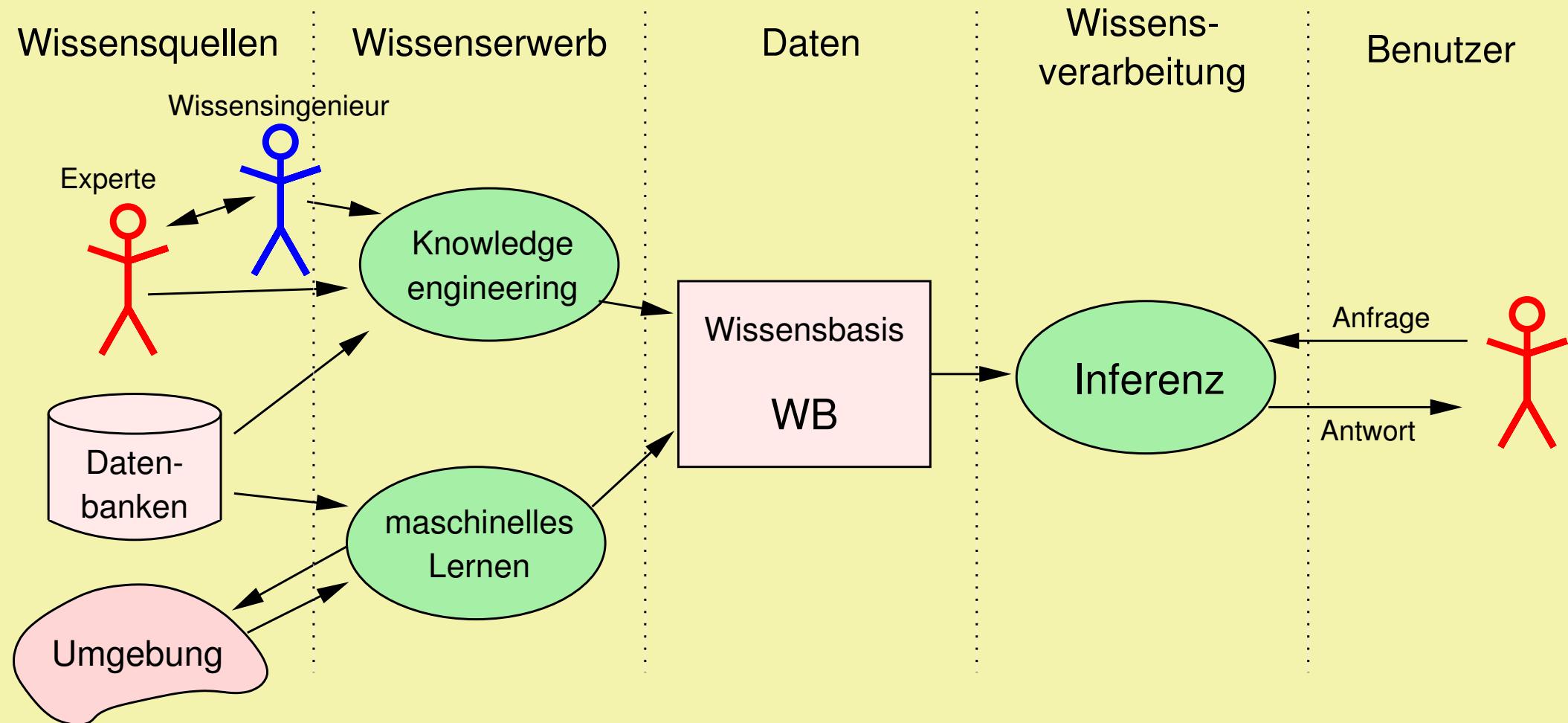
↳ „alle Computerprogramme lösbar wett“

Knowledge Engineering

Wissensquelle z.B. Datenbanken, menschl. Experten

Wissensingenieur (engl. knowledge engineer)

Maschinelles Lernen u.a. aktive Exploration durch Agenten



Struktur eines klassischen wissensverarbeitenden Systems.

Trennung von Wissen und Inferenz hat Vorteile:

- Inferenz ist anwendungsunabhängig (Beisp. med. XPS)
- Wissen kann deklarativ gespeichert werden.

Darstellung des Wissens mit formaler Sprache:

- Aussagenlogik
- Prädikatenlogik erster Stufe (kurz: PL1).
- probabilistische Logik,
- Fuzzylogik
- Entscheidungsbäume

Kapitel 2

Aussagenlogik

wenn es regnet ist die Straße nass.

Etwas formaler geschrieben ergibt sich

es regnet \Rightarrow die Straße ist nass.

Syntax

Definition 2.2 Sei $Op = \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)\}$ die Menge der logischen Operatoren und Σ eine Menge von Symbolen mit $Op \cap \Sigma \cap \{w, f\} = \emptyset$. Σ heißt **Signatur** und seine Elemente sind die **Aussagevariablen**. Die Menge der Aussagenlogischen Formeln wird nun rekursiv definiert:

- w und f sind (atomare) Formeln.
- Alle Aussagevariablen, das heißt alle Elemente von Σ sind (atomare) Formeln.
- Sind A und B Formeln, so sind auch $\neg A$, (A) , $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ Formeln.

Definition 2.4 Man liest die Operatoren und Symbole folgendermaßen:

w	“wahr”
f	“falsch”
$\neg A$	“nicht A ”
$\neg A$	(Negation)
$\neg A$	“nicht A ”
$A \wedge B$	(Negation)
$A \wedge B$	“ A und B ”
$A \vee B$	(Konjunktion)
$A \vee B$	“ A oder B ”
$A \Rightarrow B$	(Disjunktion)
$A \Rightarrow B$	“wenn A dann B ”
$A \Leftrightarrow B$	(Implikation)
$A \Leftrightarrow B$	“ A genau dann, wenn B ”
$A \Leftrightarrow B$	(Äquivalenz)

AND NOT

falls Delegung nicht existiert, ist sie nicht wahr

I	A	$\neg A$
1	w	w
2	w	f
3	f	w
4	w	f
5	f	w
6	f	f
7	f	f
8	f	f

Mit $\Sigma = \{A, B, C\}$ sind zum Beispiel

$$A \wedge B, \quad A \wedge B \wedge C, \quad A \wedge A \wedge A, \quad C \wedge B \vee A, \quad (\neg A \wedge B) \Rightarrow (\neg C \vee A)$$

Formeln. Auch $((A)) \vee B$ ist eine syntaktisch korrekte Formel.

Die so definierten Formeln sind bisher rein syntaktische Gebilde ohne Bedeutung.
Es fehlt noch die Semantik.

Semantik

Ist die Formel

$$A \wedge B$$

wahr?

Definition 2.6 Eine Abbildung $I : \Sigma \rightarrow \{w, f\}$, die jeder Aussagevariablen einen Wahrheitswert zuordnet, heißt **Belegung** oder **Interpretation** oder auch **Welt**.

Jede aussagenlogische Formel mit n verschiedenen Variablen hat 2^n verschiedene Belegungen

Wahrheitstabelle

Y

A	B	(A)	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
w	w	w	f	w	w	w	w
w	f	w	f	f	w	f	f
f	w	f	w	f	w	w	f
f	f	f	w	f	f	w	w

Die leere Formel ist unter allen Belegungen wahr.

Operatorprioritäten: \neg , \wedge , \vee , \Rightarrow , \Leftrightarrow .

Definition 2.8 Zwei Formeln F und G heißen semantisch äquivalent, wenn sie für alle Belegungen die gleichen Wahrheitswerte annehmen. Man schreibt $F \equiv G$.

Metasprache: natürlichen Sprache, z.B. “ $A \equiv B$ ”

Objektsprache: Logik, z.B. “ $A \Leftrightarrow B$ ”

Satz 2.2 Die Operationen \wedge , \vee sind kommutativ und assoziativ und es gilt:

$$\neg A \vee B \Leftrightarrow A \Rightarrow B \quad (\text{Implikation})$$

$$A \Rightarrow B \Leftrightarrow \neg B \Rightarrow \neg A \quad (\text{Kontraposition})$$

$$(A \Rightarrow B) \wedge (B \Rightarrow A) \Leftrightarrow A \Leftrightarrow B \quad (\ddot{\text{A}}\text{quivalenz})$$

$$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B \quad (\text{De Morgan'sche Regeln})$$

$$\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$$

$$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C) \quad (\text{Distributivgesetze})$$

$$A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$$

$$A \vee \neg A \Leftrightarrow w \quad (\text{Tautologie})$$

$$A \wedge \neg A \Leftrightarrow f \quad (\text{Widerspruch})$$

$$A \vee f \Leftrightarrow A$$

$$A \vee w \Leftrightarrow w$$

$$A \wedge f \Leftrightarrow f$$

$$A \wedge w \Leftrightarrow A$$

Beweis: nur erste Äquivalenz:

A	B	$\neg A$	$\neg A \vee B$	$A \Rightarrow B$	$(\neg A \vee B) \Leftrightarrow (A \Rightarrow B)$
w	w	f	w	w	w
w	f	f	f	f	w
f	w	w	w	w	w
f	f	w	w	w	w

Die Beweise für die anderen Äquivalenzen laufen analog und werden dem Leser zur □

Übung empfohlen (Aufgabe ??).

Varianten der Wahrheit

Je nachdem, in wievielen Welten eine Formel wahr ist, teilt man die Formeln ein in folgende Klassen.

Definition 2.10 Eine Formel heißt

- **erfüllbar**, falls sie bei mindestens einer Belegung wahr ist.
- **allgemeingültig** oder **wahr**, falls sie bei allen Belegungen wahr ist. Wahre Formeln heißen auch **Tautologien**.
- **unerfüllbar**, falls sie bei keiner Belegung wahr ist.

Jede erfüllende Belegung einer Formel heißt **Modell**.

Offenbar ist die Negation jeder allgemeingültigen Formel unerfüllbar. Die Negation einer erfüllbaren aber nicht allgemeingültigen Formel F ist erfüllbar.

Beweisverfahren

Aus der **Wissensbasis** WB , folgt eine Formel Q ? $Q = \text{know-how}$

Definition 2.12 Eine Formel Q **folgt** aus einer Formel WB , wenn jedes Modell von WB auch ein Modell von Q ist. Man schreibt dann $WB \models Q$.

Q ist logische Konsequenz von WB

- semantischer Begriff!
- Jede Formel wählt aus der Menge aller Belegungen eine Teilmenge als ihre Modelle aus.
- Tautologien wie zum Beispiel $A \vee \neg A$ schränken die Zahl der erfüllenden Belegungen nicht ein, denn ihre Aussage ist leer. *- alle Belegungen von WB für die das WD wahr machen müssen*
- Die leere Formel ist in allen Belegungen wahr.
- Für jede Tautologie T gilt $\emptyset \models T$, kurz $\models T$.

Wahrheitstabelle der Implikation:

A	B	$A \Rightarrow B$
w	w	w
w	f	f
f	w	w
f	f	w

Die Implikation $A \Rightarrow B$ ist immer wahr, außer bei der Belegung $A \mapsto w, B \mapsto f$. Nehmen wir an, dass $A \models B$ gilt. Das heißt, dass für jede Belegung, die A wahr macht, auch B wahr ist. Die zweite Zeile der Wahrheitstabelle kommt damit gar nicht vor. Also ist $A \Rightarrow B$ wahr, das heißt $A \Rightarrow B$ ist eine Tautologie. Damit ist eine Richtung des wichtigen folgenden Satzes gezeigt.

„ist Tautologie“ (eigentlich steht links daneben
die Lese-Farbe)

$$\models WB \Rightarrow Q$$

Satz 2.4 (Deduktionstheorem)

$WB \Rightarrow Q$.

$WB \models Q$ gilt genau dann wenn \models

$A \in B; B \subseteq A$

Wenn WB impliziert Q eine Tautologie ist

- Die Wahrheitstafelmethode ist ein **Beweisverfahren** für die Aussagenlogik!
- Nachteil: im Worst-Case sehr lange Rechenzeit (2^n Belegungen)

Wenn eine Formel Q aus WB folgt, so ist nach dem Deduktionstheorem $WB \Rightarrow Q$ eine Tautologie. Also ist die Negation $\neg(WB \Rightarrow Q)$ unerfüllbar. Wegen

$$\neg(WB \Rightarrow Q) \equiv \neg(\neg WB \vee Q) \equiv WB \wedge \neg Q$$

ist damit auch $WB \wedge \neg Q$ unerfüllbar. Also:

Satz 2.6 (Widerspruchsbeweis) $WB \models Q$ gilt genau dann wenn $WB \wedge \neg Q$ unerfüllbar ist.

Um zu zeigen, dass die zu beweisende Anfrage Q aus der Wissensbasis WB folgt, kann man also die negierte Anfrage $\neg Q$ zur Wissensbasis hinzufügen und daraus einen Widerspruch ableiten.

$$\begin{array}{c} \cancel{| WB \Rightarrow Q } \\ \vdash \\ \text{⊕ } \neg Q \text{ nicht erfüllbar} \end{array}$$

Anwendungen:

- in der Mathematik
- in vielen automatischen Beweiskalkülen, u.a. Resolution, PROLOG

$$A \wedge B \vdash C \xrightarrow{\text{Löse-Kalkül (Lösung Q)}} B \vdash C$$

Löse-Kalkül (Lösung Q)

Ableitung: syntaktische Manipulation der Formeln WB und Q durch Anwendung von Inferenzregeln mit dem Ziel, diese so stark zu vereinfachen, dass man am Ende sofort erkennt, dass $WB \models Q$. Man schreibt dann $WB \vdash Q$.

Kalkül: syntaktisches Beweisverfahren (Ableiten).

$WB \vdash Q$ („gib ihm 500 € als Bankkontostand“)

$\rightarrow WB \vdash Q$ („Kunde hat ≥ 500 € im Konto“)

Formeln umbauen

„Realität“

„Computerprogramm“

Ziel = Verbindung von Programm zur Realität herstellen

Korrektheit und Vollständigkeit!

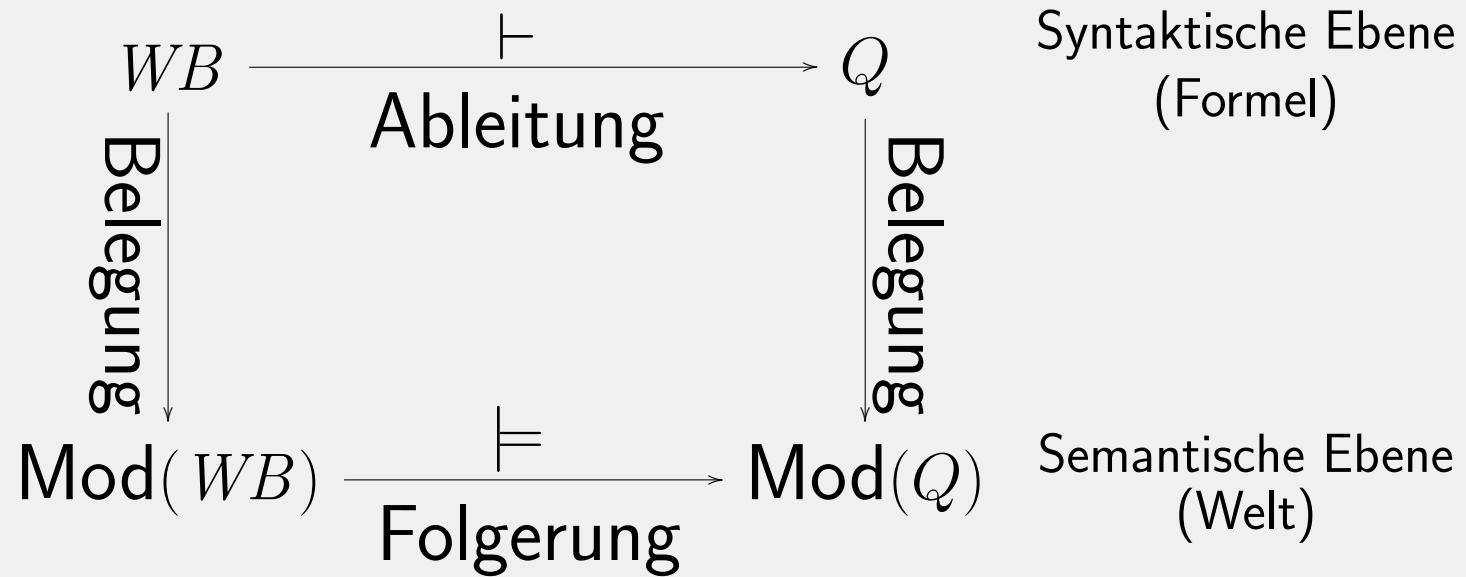
Definition 2.14 Ein Kalkül heißt **korrekt**, wenn jede abgeleitete Aussage auch semantisch folgt, das heißt, wenn für Formeln WB und Q gilt

wenn $WB \vdash Q$ dann $WB \models Q$.

Ein Kalkül heißt **vollständig**, wenn alle semantischen Folgerungen abgeleitet werden können, das heißt, wenn für Formeln WB und Q gilt

wenn $WB \models Q$ dann $WB \vdash Q$. „alle“

Syntaktische Ableitung und semantische Folgerung



$Mod(X)$ steht für die Menge der Modelle einer Formel

Um die automatischen Beweisverfahren möglichst einfach zu halten, arbeiten diese meist auf Formeln in konjunktiver Normalform.

Definition 2.16 Eine Formel ist in **konjunktiver Normalform (KNF)** genau dann, wenn sie aus einer **Konjunktion**

$$K_1 \wedge K_2 \wedge \dots \wedge K_m$$

$K = \text{Klausel}$

von Klauseln besteht. Eine Klausel K_i besteht aus einer **Disjunktion**

$$(L_{i1} \vee L_{i2} \vee \dots \vee L_{in_i})$$

von **Literalen**. Ein **Literal** schließlich ist eine Variable (positives Literal) oder eine negierte Variable (negatives Literal). Was ist Literal? \rightarrow Element von $\Sigma \rightarrow \{A, B, C, \bar{A}, \bar{B}, \bar{C}\}$

Die Formel $(A \vee B \vee \neg C) \wedge (A \vee B) \wedge (\neg B \vee \neg C)$ ist in konjunktiver Normalform. Die konjunktive Normalform stellt keine Einschränkung der Formelmenge dar, denn es gilt

Satz 2.8 Jede aussagenlogische Formel lässt sich in eine äquivalente konjunktive Normalform transformieren.

Beispiel: Wir bringen $A \vee B \Rightarrow C \wedge D$ in konjunktive Normalform, indem wir die Äquivalenzen aus Satz 2.2 anwenden:

$$A \vee B \stackrel{\gamma\vee}{\Rightarrow} C \wedge D$$

* ∨

$$\begin{aligned}
 &\equiv \neg(A \vee B) \vee (C \wedge D) && \bullet \quad (\text{Implikation}) \\
 &\equiv (\neg A \wedge \neg B) \vee (C \wedge D) && \quad (\text{de Morgan}) \\
 &\equiv (\neg A \vee (C \wedge D)) \wedge (\neg B \vee (C \wedge D)) && \quad (\text{Distributivgesetz}) \\
 &\equiv ((\neg A \vee C) \wedge (\neg A \vee D)) \wedge ((\neg B \vee C) \wedge (\neg B \vee D)) && \quad (\text{Distributivgesetz}) \\
 &\equiv (\neg A \vee C) \wedge (\neg A \vee D) \wedge (\neg B \vee C) \wedge (\neg B \vee D) && \quad (\text{Assoziativgesetz})
 \end{aligned}$$

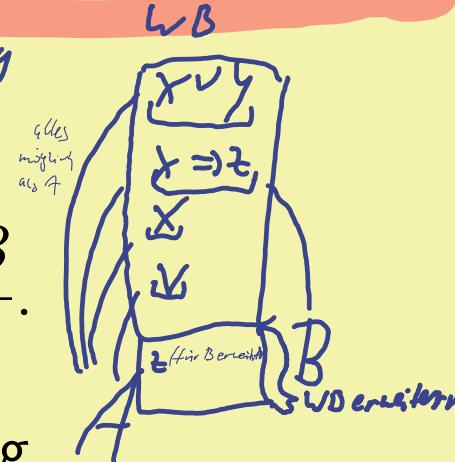
Klausur: Formel in konjunktive NF bringen!

Beweiskalkül: Modus Ponens

Pattern-Matching

keine
Formel {

$$\text{Platzhalter; nicht } \in \{ \quad \frac{A, \quad A \Rightarrow B}{B}.$$



Modus Ponens ist korrekt aber nicht vollständig.

Resolution (1965)

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C}. \quad (2.1)$$

bzw.

$$\frac{A \vee B, \quad B \Rightarrow C}{A \vee C}.$$

- Die abgeleitete Klausel wird Resolvente genannt.
- Resolution ist eine Verallgemeinerung des Modus Ponens.

Beispiel: $B \wedge \neg B \vdash \{\}$

allgemeine Resolutionsregel:

$$\frac{(A_1 \vee \dots \vee A_m \vee B), (\neg B \vee C_1 \vee \dots \vee C_n)}{(A_1 \vee \dots \vee A_m \vee C_1 \vee \dots \vee C_n)}. \quad (2.2)$$

Man nennt die Literale B und $\neg B$ **komplementär**.

Satz 2.10 Der Resolutionskalkül zum Beweis der Unerfüllbarkeit von Formeln in konjunktiver Normalform ist korrekt und vollständig.

die Wissensbasis WB muss widerspruchsfrei sein!

Andernfalls lässt sich aus WB alles herleiten (siehe Aufgabe ??).

Beispiel: Das Logikrätsel Nr. 7 aus Berrondo mit dem Titel **A charming English family** lautet:

Nachdem ich sieben Jahre lang mit glänzendem Erfolg Englisch studiert habe, muss ich zugeben, dass ich, wenn ich Engländer englisch sprechen höre, vollkommen perplex bleibe. Nun habe ich neulich, von edlen Gefühlen bewegt, drei Anhalter, Vater, Mutter und Tochter, mitgenommen, die, wie ich schnell begriff, Engländer waren und folglich nur Englisch sprachen. Bei jedem ihrer Sätze zögerte ich zwischen zwei möglichen Bedeutungen. Sie sagten mir folgendes (der zweite Sinn steht in Klammern): Der Vater: "Wir fahren bis nach **Spanien** (wir kommen aus **Newcastle**)."
Die Mutter: "Wir fahren nicht nach Spanien und kommen aus Newcastle (wir haben in **Paris** angehalten und fahren nicht nach Spanien)."
Die Tochter: "Wir kommen nicht auch Newcastle (wir haben in Paris angehalten)."
What about this charming English family?

drei Schritte:

- Formalisierung (oft sehr schwierig)

- Transformation in Normalform
- Beweis (oft sehr schwierig)

praktisches Üben hier wichtig! (siehe Aufgaben ??–??).

$$(S \vee N) \wedge [(\neg S \wedge N) \vee (P \wedge \neg S)] \wedge (\neg N \vee P).$$

Ausklammern von $\neg S$ in der mittleren Teilformel bringt die Formel in einem Schritt in KNF.

$$WB \equiv (S \vee N)_1 \wedge (\neg S)_2 \wedge (P \vee N)_3 \wedge (\neg N \vee P)_4.$$

Nun starten wir den Resolutionsbeweis (ohne Anfrage Q).

$$\begin{aligned} \text{Res}(1,2) &: (N)_5 \\ \text{Res}(3,4) &: (P)_6 \\ \text{Res}(1,4) &: (S \vee P)_7 \end{aligned}$$

Leere Klausel nicht ableitbar. Also ist WB widerspruchsfrei.

Um zu zeigen, dass $\neg S$ gilt, fügen wir die Klausel $(S)_7$ als negierte Anfrage zur Klauselmenge hinzu.

Res(2,7) : ()₈

Es gilt also $\neg S \wedge N \wedge P$.

Die “charming English family” kommt offenbar aus Newcastle, hat in Paris angehalten und fährt nicht nach Spanien.

Beispiel: Das Logikrätsel Nr. 28 aus [Berrondo](#) mit dem Titel **Der Hochsprung** lautet:

Drei junge Mädchen üben Hochsprung für die Sportprüfung im Abitur. Die Stange wurde bei 1.20 m befestigt. "Ich wette", sagt das erste zum zweiten Mädchen, "dass mir mein Sprung gelingen wird, wenn, und nur dann, wenn du versagst." Wenn das zweite junge Mädchen das gleiche zu dem dritten sagt, welches wiederum das gleiche zu dem ersten sagt, wäre es dann möglich, dass keins von den dreien seine Wette verliert?

Wir zeigen per Resolutionsbeweis, dass nicht alle drei Mädchen ihre Wette gewinnen können.

Formalisierung:

Der Sprung des ersten Mädchens gelingt: A , Wette des ersten Mädchens: $(A \Leftrightarrow \neg B)$
 der Sprung des zweiten Mädchens gelingt: B , Wette des zweiten Mädchens: $(B \Leftrightarrow \neg C)$
 der Sprung des dritten Mädchens gelingt: C . Wette des dritten Mädchens: $(C \Leftrightarrow \neg A)$

Behauptung: Es können nicht alle drei die Wette gewinnen:

$$Q \equiv \neg((A \Leftrightarrow \neg B) \wedge (B \Leftrightarrow \neg C) \wedge (C \Leftrightarrow \neg A))$$

Per Resolution ist nun zu zeigen, dass $\neg Q$ unerfüllbar ist.

Transformation in KNF: Wette des ersten Mädchens:

$$(A \Leftrightarrow \neg B) \equiv (A \Rightarrow \neg B) \wedge (\neg B \Rightarrow A) \equiv (\neg A \vee \neg B) \wedge (A \vee B)$$

Die Wetten der beiden anderen Mädchen werden analog transformiert und man erhält als negierte Behauptung

$$\neg Q \equiv (\neg A \vee \neg B)_1 \wedge (A \vee B)_2 \wedge (\neg B \vee \neg C)_3 \wedge (B \vee C)_4 \wedge (\neg C \vee \neg A)_5 \wedge (C \vee A)_6$$

Daraus wird nun mit Resolution die leere Klausel abgeleitet:

$$\text{Res}(1,6) : (C \vee \neg B)_7$$

$$\text{Res}(4,7) : (C)_8$$

$$\text{Res}(2,5) : (B \vee \neg C)_9$$

$$\text{Res}(3,9) : (\neg C)_{10}$$

$$\text{Res}(8,10) : ()$$

Damit ist die Behauptung gezeigt.

Hornklauseln

Eine Klausel in konjunktiver Normalform enthält positive und negative Literale und lässt sich daher darstellen in der Form

nicht definite Klausel ist eine

$(\neg A_1 \vee \dots \vee \neg A_m \vee B_1 \vee \dots \vee B_n)$ *Hornklausel*

Diese Klausel lässt umformen in

$$A_1 \wedge \dots \wedge A_m \Rightarrow B_1 \vee \dots \vee B_n.$$

Beispiele:

“Wenn das Wetter schön ist und Schnee liegt, gehe ich Skifahren oder ich gehe arbeiten.” (nicht definite Klausel)

, pretty useless statement

“Wenn das Wetter schön ist und Schnee liegt, gehe ich Skifahren.”.
(definite Klausel)

= Hornklausel (maximal 1 positives Literal)



What disease formed the Borkumseel?

$(\neg A_1 \vee \dots \vee \neg A_m \vee B)$ oder $(\neg A_1 \vee \dots \vee \neg A_m)$ oder B

beziehungsweise (äquivalent)

10

$$A_1 \wedge \dots \wedge A_m \Rightarrow B \quad \text{oder} \quad A_1 \wedge \dots \wedge A_m \Rightarrow f \quad \text{oder} \quad B.$$

heißen **Hornklauseln** (nach ihrem Erfinder). Eine Klausel mit nur einem positiven Literal heißt **Fakt**. Bei Klauseln mit negativen und einem positiven Literal wird das positive Literal **Kopf** genannt.

Zum besseren Verständnis der Darstellung von Hornklauseln möge der Leser die in der Definition verwendeten Äquivalenzen herleiten (Aufgabe ??).

Beispiel:

Wissensbasis:

- $(Wetter_schön)_1$
- $(Schneefall)_2$
- $(Schneefall \Rightarrow Schnee)_3$
- $(Wetter_schön \wedge Schnee \Rightarrow Skifahren)_4$

Gilt **Skifahren**?

Inferenzregel (verallgemeinerter Modus Ponens):

$$\frac{A_1 \wedge \dots \wedge A_m, \quad A_1 \wedge \dots \wedge A_m \Rightarrow B}{B}.$$

Beweis für **Skifahren**:

$$\begin{aligned} \text{MP}(2,3) : & \quad (Schnee)_5 \\ \text{MP}(1,5,4) : & \quad (Skifahren)_6 \end{aligned}$$

Modus Ponens ist vollständig für aussagenlogische Hornklauseln

forward chaining: startet bei den Fakten und leitet die Anfrage her

backward chaining: mit der Anfrage starten und rückwärts arbeiten, bis die Fakten erreicht sind

Prolog macht backward chaining

Ch. 5

SLD-Resolution

“Selection rule driven linear resolution for definite clauses”.

Beispiel, ergänzt durch die negierte Anfrage $(Skifahren \Rightarrow f)$:

$(Wetter_schön)_1$
 $(Schneefall)_2$
 $(Schneefall \Rightarrow Schnee)_3$
 $(Wetter_schön \wedge Schnee \Rightarrow Skifahren)_4$
 $(Skifahren \Rightarrow f)_5$

SLD-Resolution:

Res(5,4) : $(Wetter_schön \wedge Schnee \Rightarrow f)_6$
Res(6,1) : $(Schnee \Rightarrow f)_7$
Res(7,3) : $(Schneefall \Rightarrow f)_8$
Res(8,2) : ()

- **linear resolution:** es wird immer mit der gerade aktuell hergeleiteten Klausel weitergearbeitet.
- Reduktion des Suchraumes.
- Literale der aktuellen Klausel werden immer der Reihe nach in fester Reihenfolge (z.B. von links nach rechts) abgearbeitet (“**Selection rule driven**”).
- Literale der aktuellen Klausel werden **Teilziele** (engl. subgoals) genannt.
- Literale der negierten Anfrage sind die **Ziele** (engl. goals).

Inferenzregel:

$$\frac{A_1 \wedge \dots \wedge A_m \Rightarrow B_1, \quad B_1 \wedge B_2 \wedge \dots \wedge B_n \Rightarrow f}{A_1 \wedge \dots \wedge A_m \wedge B_2 \wedge \dots \wedge B_n \Rightarrow f}.$$

SLD-Resolution und PROLOG

- Der Beweis (Widerspruch) ist gefunden, wenn die Liste der Teilziele der aktuellen Klausel (**goal stack**) leer ist.
- Gibt es zu einem Teilziel $\neg B_i$ keine Klausel mit einem komplementären Literal B_i als Klauselkopf, so terminiert der Beweis und es kann kein Widerspruch gefunden werden.
- PROLOG-Programme bestehen aus prädikatenlogischen Hornklauseln
- Abarbeitung erfolgt mittels SLD-Resolution.

Berechenbarkeit und Komplexität

- Wahrheitstafelmethode bestimmt nach endlicher Zeit alle Modelle.
- Die Mengen der unerfüllbaren, der erfüllbaren und der allgemeingültigen Formeln sind entscheidbar.
- $T(n) = O(2^n)$
- Optimierung: **Semantische Bäume**, im Worst-Case auch exponentiell.
- Bei der Resolution wächst die Zahl der abgeleiteten Klauseln im Worst-Case exponentiell mit der Zahl der Klauseln.
- S. Cook: 3-Sat-Problem ist NP-vollständig
- 3-Sat ist die Menge aller KNF-Formeln deren Klauseln genau drei Literale haben.
- Für Hornklauseln: Zeit für Erfüllbarkeitstest wächst nur linear mit der Zahl der Literale in der Formel.

Anwendungen und Grenzen

- Digitaltechnik
- Verifikation digitaler Schaltungen
- Generierung von Testmustern
- einfache KI-Anwendungen: diskrete Variablen, wenige Werte, keine Relationen zw. Var.
- probabilistische Logik (Kapitel 7) verwendet Aussagenlogik, modelliert Unsicherheit
- Fuzzylogik, erlaubt unendlich viele Wahrheitswerte.

Kapitel 3

Prädikatenlogik erster Stufe

Aussage:

Roboter 7 befindet sich an xy-Position (35,79)

aussagenlogische Variable:

Roboter_7_befindet_sich_an_xy-Position_(35,79)

100 Roboter auf Gitter mit 100×100 Punkten.

$$\Rightarrow 100 \cdot 100 \cdot 100 = 1\,000\,000 = 10^6 \text{ Variablen}$$

Viel zu viel; man müsst für
 ↴ jeden Roboter für alle 100×100 Punkte
 sagen, welche Stelle True ist, Rest wäre
FALSE

Relation

Roboter A steht weiter rechts als Roboter B.

$(100 \cdot 99)/2 = 4950$ geordnete Paare aus x -Werten.

10^4 Formeln der Art:

$$\begin{aligned}
 & \text{Roboter_7_steht_weiter_rechts_als_Roboter_12} \Leftrightarrow \\
 & \quad \text{Roboter_7_befindet_sich_an_xy_Position_}(35,79) \\
 & \quad \wedge \text{Roboter_12_befindet_sich_an_xy_Position_}(10,93) \\
 & \vee \dots
 \end{aligned}$$

Implication
 ↴

mit $(10^4)^2 \cdot 0.495 = 0.495 \cdot 10^8$ Alternativen auf der rechten Seite.

in Logik braucht man viele Variablen, da eine Variable nur
 wahr oder falsch sein kann

Prädikatenlogik erster Stufe:

$Position(\text{nummer}, \text{xPosition}, \text{yPosition})$

*u steht weiter rechts
von v*

$\forall u \forall v \text{ Steht_weiter_rechts}(u, v) \Leftrightarrow$

$\exists x_u \exists y_u \exists x_v \exists y_v \ Position(u, x_u, y_u) \wedge Position(v, x_v, y_v) \wedge x_u > x_v,$

Syntax

Termen, z.B.: $f(\sin(\ln(3)), \exp(x))$

Definition 3.2 Sei V eine Menge von Variablen, K eine Menge von Konstanten und F eine Menge von Funktionssymbolen mit $V \cap K \cap F = \emptyset$. Die Menge der Terme wird rekursiv definiert:

- Alle Variablen und Konstanten sind (atomare) Terme.
- Sind t_1, \dots, t_n Terme und f ein n -stelliges Funktionssymbol, so ist auch $f(t_1, \dots, t_n)$ ein Term.

Beispiel:

$$V = \{x, y, z\}$$

$$K = \{a, b\}$$

$$F = \{y_0, h_0, h_1\}$$

Terme:

1) x, y, z, a, b (atomare Terme)

2) $h_0(x)$

3) $h_1(x, y)$

4) $h_0/h_1(x, y)$

Es gibt noch viele weitere
Terme!

$g()$ & exotisches Funktionsymbol

$g(,)$ & Etwasstelliges Funktionssymbol

Definition 3.4 Sei P eine Menge von Prädikatssymbolen. **Prädikatenlogische Formeln** sind wie folgt aufgebaut:

- Sind t_1, \dots, t_n Terme und p ein n -stelliges Prädikatssymbol, so ist $p(t_1, \dots, t_n)$ eine (atomare) Formel.
 - Sind A und B Formeln, so sind auch $\neg A$, (A) , $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ Formeln.
 - Ist x eine Variable und A eine Formel, so sind auch $\forall x A$ und $\exists x A$ Formeln.
 \forall ist der All- und \exists der Existenzquantor.
-
- $p(t_1, \dots, t_n)$ und $\neg p(t_1, \dots, t_n)$ heißen Literale. Spezialbegriffe
 - Formeln, bei denen jede Variable im Wirkungsbereich eines Quantors ist, heißen **Aussagen** oder **geschlossene Formeln**. Variablen, die nicht im Wirkungsbereich eines Quantors sind, heißen **freie Variablen**.
 - Die Definitionen 2.16 (KNF) und 2.18 (Hornklausel) gelten für Formeln aus prädikatenlogischen Literalen analog.

$$\forall x \varphi(x)$$

$$\exists y \varphi(x)$$

Beispiel:

$$V = \{x, y, z, \Delta\}$$

$$L = \{a, b\}$$

$$F = \{g_1, h_1, h_2\}$$

$$D = \{p_2, q_1\}$$

Formeln:

1) $q_1(h_2(x, y)) \rightarrow$ atomare Formel (siehe 3.4; Prädikatsymbol \rightarrow Formel)

2) $\exists y \Delta_2(\Delta_1(x, y)) \rightarrow$ Term, \exists keine Formel; Prädikatsymbol fehlt

3) $\exists y p_2(q_1(x), a) \rightarrow q_1$ ist keine Formel!

4) $\exists y p_2(q_1(x), y) \vee \rightarrow$ atomare Formel

5) $\exists a \neg q_1(h_2(x, y)) \rightarrow$ ist Formel, aber nicht atomar, da sie aus Formel 1 hergeleitet wurde

6) $\forall x p_2(x, a) \rightarrow$ Formel, nicht atomar, keine Aussage

7) $\exists x \Delta_1(p_2(x, a)) \rightarrow$ " " , keine Aussage

8) $\exists x p_2(x, a) \rightarrow$ " , ist Aussage, da für alle Variable (hier nur x) im Einflussbereich eines Quantors stehen

9) $\forall x p_2(x, y) \rightarrow$ keine Aussage, da y eine freie Variable ist

Beispiele:

Formel	Beschreibung
$\forall x \ frosch(x) \Rightarrow grün(x)$	Alle Frösche sind grün
$\forall x \ frosch(x) \wedge braun(x) \Rightarrow groß(x)$	Alle braunen Frösche sind groß
$\forall x \ mag(x, kuchen)$	Jeder mag Kuchen
$\neg \forall x \ mag(x, kuchen)$	Nicht jeder mag Kuchen
$\neg \exists x \ mag(x, kuchen)$	Keiner mag Kuchen
$\exists x \ \forall y \ mag(y, x)$	Es gibt etwas, das jeder mag
$\exists x \ \forall y \ mag(x, y)$	Es gibt jemanden, der alles mag
$\forall x \ \exists y \ mag(y, x)$	Jedes Ding wird von jemandem geliebt
$\forall x \ \exists y \ mag(x, y)$	Jeder mag etwas
$\forall x \ kunde(x) \Rightarrow mag(bob, x)$	Bob mag jeden Kunden

$$\exists x \ kunde(x) \wedge \ mag(x, bob)$$

Es gibt einen Kunden der Bob
mag

$$\exists x \ baecker(x) \wedge \forall y \ kunde(y) \Rightarrow \ mag(x, y)$$

Es gibt einen Bäcker der all sei-
ne Kunden mag

Semantik

Definition 3.6 Eine **Belegung** \mathbb{B} oder **Interpretation** ist definiert durch

- Eine Abbildung von der Menge der Konstanten und Variablen $K \cup V$ auf eine Menge W von Objekten aus der Welt.
- Eine Abbildung von der Menge der Funktionssymbole auf die Menge der Funktionen der Welt. Jedem n -stetigen Funktionssymbol wird eine n -stellige Funktion zugeordnet.
- Eine Abbildung von der Menge der Prädikatssymbole auf die Menge der Relationen der Welt. Jedem n -stetigen Prädikatssymbol wird eine n -stellige Relation zugeordnet.

Beispiel:

$$V = \{x, y, z, \textcolor{brown}{B_0}, \textcolor{brown}{B_1}\}$$

$$K = \{a_1, b\}$$

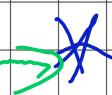
$$F = \{g_1, h_1, h_2\}$$

$$P = \{p_1, q_1\}$$

B_0

B_1

\hookrightarrow (Abbildung Konst. u. Variablen)



4

2

Funktionen (Abbildung Fkt)

f_2

str₁

Haus streichen

Relationen (Abbildung v. Prädikatsymbolen)

$$Y = \{(2, 7), (3, 7), (3, 2)\}$$

$$\text{istgrün} = \{(;), (\text{t})\}$$

istgrün

Alle Elemente auf
d. linken Seite müssen
etwas Objekt bezeichnen
wenden, B_0 wäre nicht
vollständig und nach Idee
Delegierung

Beispiel: Konstanten: c_1, c_2, c_3 , zweistelliges Funktionssymbol “plus”, zweistelliges Prädikatssymbol “gr”.

$$F \equiv gr(\text{plus}(c_1, c_3), c_2)$$

Wähle Belegung:

$$\mathbb{B}_1 : c_1 \mapsto 1, c_2 \mapsto 2, c_3 \mapsto 3, \quad \text{plus} \mapsto +, \quad gr \mapsto >$$

Damit wird die Formel abgebildet auf

$$1 + 3 > 2, \quad \text{bzw. nach Auswertung} \quad 4 > 2$$

Größer-Relation G auf $\{1, 2, 3, 4\}$:

$$G = \{(4, 3), (4, 2), (4, 1), (3, 2), (3, 1), (2, 1)\}.$$

Da $(4, 2) \in G$, ist F unter der Belegung \mathbb{B}_1 wahr.

$$\mathbb{B}_2 : c_1 \mapsto 2, c_2 \mapsto 3, c_3 \mapsto 1, \quad plus \mapsto -, \quad gr \mapsto >,$$

ergibt

$$2 - 1 > 3, \quad \text{bzw.} \quad 1 > 3.$$

$(1, 3)$ ist nicht in G enthalten.

Die Wahrheit einer Formel in PL1 hängt also von der Belegung ab.

Definition 3.8

- Eine atomare Formel $p(t_1, \dots, t_n)$ ist **wahr** unter der Belegung \mathbb{B} , wenn nach Belegung und Auswertung aller Terme t_1, \dots, t_n und Belegung des Prädikates p durch die n -stellige Relation r gilt

$$(\mathbb{B}(t_1), \dots, \mathbb{B}(t_n)) \in r.$$

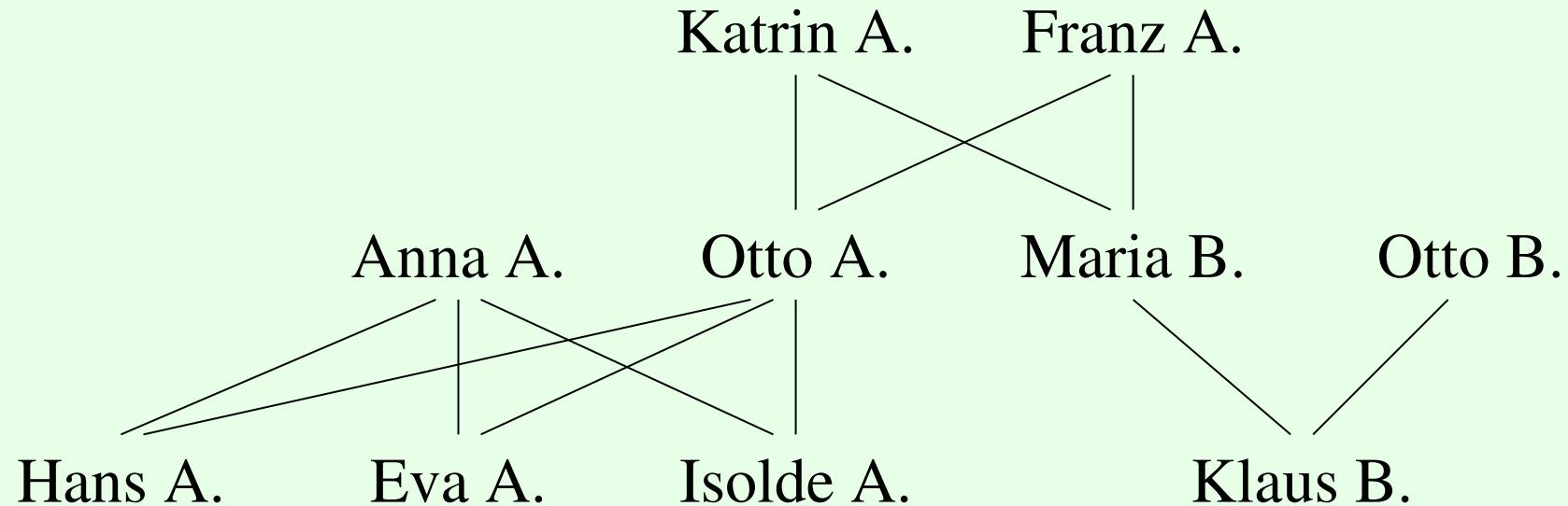
(1)

- Die Wahrheit von quantorenfreien Formeln ergibt sich aus der Wahrheit der atomaren Formeln – wie in der Aussagenlogik – über die Semantik der logischen Operatoren. (2)
- Eine Formel $\forall x F$ ist wahr unter der Belegung \mathbb{B} genau dann wenn sie bei beliebiger Änderung der Belegung für die Variable x wahr ist. (3)
- Eine Formel $\exists x F$ ist wahr unter der Belegung \mathbb{B} genau dann wenn es für die Variable x eine Belegung gibt, welche die Formel wahr macht. (4)

Definitionen zur semantischen Äquivalenz von Formeln, für erfüllbar, wahr, unerfüllbar und Modell, semantische Folgerung werden aus der Aussagenlogik übernommen.

Satz 3.2 Die Sätze Satz 2.4 (Deduktionstheorem), und 2.6 (Widerspruchsbeweis) gelten analog auch für PL1.

- ① getrennte Formel: $p_2(c_1, x)$ nach ^{längere Belegung so verändert} Belegung $\rightarrow p_2(x, \overbrace{c_0, c_1})$
 \square c_0, c_1 und y kommen nicht in Relation \rightarrow Formel $p_2(y, x)$ ist nicht wahr!
-
- ② $p_2(x, y) \wedge q_1(x)$ (nur Annahme, nicht unbedingt so, wenn $p_2(x, y)$ ist wahr
 und $q_1(x)$ falsch, dann komplexe Formel falsch)
-
- ③ $\exists y p_2(x, y) \wedge q_1(x) \Rightarrow$ hier muss überprüft werden, ob die Formel wahr ist, wenn y für jedes Objekt eingesetzt wird.
 hier: $p_2(x, *) \wedge q_1(x)$, $p_2(x, \overbrace{c_0, c_1}) \wedge q_1(x)$
 $p_2(x, 4) \wedge q_1(4)$, $p_2(x, 2) \wedge q_1(x)$
-
- ④ Achtung bei ③ schreibt, nur q_1 kann wahr annehmen, wenn $x = y$ ist $\exists y p_2(x, y) \wedge q_1(x)$

Beispiel: Ein Stammbaum:

Relation: (Kind, Mutter, Vater)

Kind = { (Otto A., Katrin A., Franz A.), (Maria B., Katrin A., Franz A.),
 (Hans A., Anna A., Otto A.), (Eva A., Anna A., Otto A.),
 (Isolde A., Anna A., Otto A.), (Klaus B., Maria B., Otto B.) }

einstellige Relation:

istWeiblich = {Katrin A., Anna A., Maria B., Eva A., Isolde A.}

Prädikate:

$\text{kind}(x, y, z)$ mit der Semantik: $L(x) = \text{Kind von } y \text{ und } z$

$$\mathbb{B}(\text{kind}(x, y, z)) = w \quad \equiv \quad (\mathbb{B}(x), \mathbb{B}(y), \mathbb{B}(z)) \in \text{Kind}.$$

Belegung:

$$\mathbb{B}(\text{otto}) = \text{Otto A.,}$$

$$\mathbb{B}(\text{eva}) = \text{Eva A.,}$$

$$\mathbb{B}(\text{anna}) = \text{Anna A.}$$

$\text{kind}(\text{eva}, \text{anna}, \text{otto})$ ist wahr!

1, NO MÜTTEL VATER!

gilt auch $\text{kind}(\text{eva}, \text{otto}, \text{anna})$?

hier nicht ohne weiter

regelmäßige Formel

$$\forall x \forall y \forall z \text{ kind}(x, \underbrace{y}_{\text{direktes Kind von } y}, z) \Leftrightarrow \text{kind}(x, \underbrace{z}_{\text{indirektes Kind von } y}, y),$$

$$\forall x \forall y \text{ nachkomme}(x, y) \Leftrightarrow \exists z \text{ kind}(x, y, z) \vee \\ (\exists u \exists v \text{ kind}(x, u, v) \wedge \text{nachkomme}(u, y)).$$

direktes Kind von y
 (indirektes Kind von y , z.B. Y als Oma)

Siehe auch Aufgabe ??!

Wissensbasis

$$\begin{aligned}
 WB \equiv & \text{weiblich(katrin)} \wedge \text{weiblich(anna)} \wedge \text{weiblich(maria)} \\
 & \wedge \text{weiblich(eva)} \wedge \text{weiblich(isolde)} \\
 & \wedge \text{kind(otto,katrin,franz)} \wedge \text{kind(maria,katrin,franz)} \\
 & \wedge \text{kind(eva,anna,otto)} \wedge \text{kind(hans,anna,otto)} \\
 & \wedge \text{kind(isolde,anna,otto)} \wedge \text{kind(klaus,maria,ottob)} \\
 & \wedge (\forall x \ \forall y \ \forall z \ \text{kind}(x, y, z) \Rightarrow \text{kind}(x, z, y)) \\
 & \wedge (\forall x \ \forall y \ \text{nachkomme}(x, y) \Leftrightarrow \exists z \ \text{kind}(x, y, z) \vee \\
 & \quad (\exists u \ \exists v \ \text{kind}(x, u, v) \wedge \text{nachkomme}(u, y))).
 \end{aligned}$$

Ist $\text{kind}(\text{eva}, \text{otto}, \text{anna})$ ableitbar?

Ist $\text{nachkomme}(\text{eva}, \text{franz})$ ableitbar?

↑ Basis für spätere Reg.

Gleichheit

Gleichheitsaxiome:

$$\begin{aligned}
 & \forall x \text{ eq}(x, x) \quad (\text{Reflexivität}) \\
 & \forall x \forall y \text{ eq}(x, y) \Rightarrow \text{eq}(y, x) \quad (\text{Symmetrie}) \\
 & \forall x \forall y \forall z \quad x = y \wedge y = z \Rightarrow x = z \quad (\text{Transitivität})
 \end{aligned} \tag{3.1}$$

$$\forall x \forall y \quad x = y \Rightarrow f(x) = f(y) \quad (\text{Substitutionsaxiom}) \tag{3.2}$$

! Nennen Sie mir die Gleichheitsaxiome!

Ersetzung von Termen

Beispiel:

$$\forall x \ x = 5 \Rightarrow x = y$$

ersetze y durch $\sin(x)$:

$$\forall x \ x = 5 \Rightarrow x = \sin(y) \quad \text{falsch!}$$

korrekt:

$$\forall x \ x = 5 \Rightarrow x = \sin(z)$$

Definition 3.10 Man schreibt $\varphi[x/t]$ für die Formel, die entsteht, wenn man in φ jedes freie Vorkommen der Variable x durch den Term t ersetzt. Hierbei dürfen in dem Term t keine Variablen vorkommen, über die in φ quantifiziert wurde. Gegebenenfalls müssen Variablen umbenannt werden, um dies sicherzustellen.

Ersetzungsoperator

Beispiel: Wird in der Formel $\forall x \ x = y$ die freie Variable y durch den Term $x + 1$ ersetzt, so ergibt sich $\forall x \ x = x + 1$. Bei korrekter Ersetzung erhält man die Formel $\forall x \ x = y + 1$ mit einer ganz anderen Semantik.

Quantoren und Normalformen

Definition 3.8:

$$\forall x \ p(x) \equiv p(a_1) \wedge \dots \wedge p(a_n)$$

für alle Konstanten $a_1 \dots a_n$ aus K .

$$\exists x \ p(x) \equiv p(a_1) \vee \dots \vee p(a_n)$$

deMorgan-Regeln:

$$\forall x \ \varphi \equiv \neg \exists \neg \varphi.$$

Beispiel:

Jeder will geliebt werden \equiv **Keiner will nicht geliebt werden**.

Definition 3.12 Eine prädikatenlogische Formel φ ist in **pränexer Normalform**, wenn gilt Quantorenstufen, wenn

- $\varphi = Q_1 x_1 \dots Q_n x_n \psi$.
- ψ ist eine quantorenfreie Formel.
- $Q_i \in \{\forall, \exists\}$ für $i = 1, \dots, n$.

$$\forall x \forall y | p(x, y) \Leftrightarrow (\forall x p(x) \wedge \exists y p(y))$$

alle Quantoren keine Quantoren

Quantoren hinter quantorenfreie Formel

Liegende Prädikate NF

Achtung:

$$\forall x \ p(x) \Rightarrow \exists x \ q(x).$$

Variable umbenennen:

$$\forall x \ p(x) \Rightarrow \exists y \ q(y)$$

Quantor nach vorne bringen:

$$\forall x \ \exists y \ p(x) \Rightarrow q(y).$$

Wie geht das mit folgender Formel?

$$(\forall x \ p(x)) \Rightarrow \exists y \ q(y) \quad (3.3)$$

Beispiel:

Konvergenz einer Folge $(a_n)_{n \in \mathbb{N}}$ gegen den Grenzwert a :

$$\forall \varepsilon > 0 \exists n_0 \in \mathbb{N} \forall n > n_0 |a_n - a| < \varepsilon$$

formal:

abs(x) für $|x|$, *a*(n) für a_n ,

minus(x, y) für $x - y$,

el(x, y) für $x \in y$, *gr*(x, y) für $x > y$:

$$\forall \varepsilon (gr(\varepsilon, 0) \Rightarrow \exists n_0 (el(n_0, \mathbb{N}) \Leftrightarrow \forall n (gr(n, n_0) \Rightarrow gr(\varepsilon, abs(minus(a(n), a)))))), \quad (3.4)$$

Implikationen eliminieren:

$$\forall \varepsilon (\neg gr(\varepsilon, 0) \vee \exists n_0 (\neg el(n_0, \mathbb{N}) \vee \forall n (\neg gr(n, n_0) \vee gr(\varepsilon, abs(minus(a(n), a)))))).$$

Quantoren nach vorne:

$$\forall \varepsilon \exists n_0 \forall n (\neg gr(\varepsilon, 0) \vee \neg el(n_0, \mathbb{N}) \vee \neg gr(n, n_0) \vee gr(\varepsilon, abs(minus(a(n), a))))$$

Reihenfolge d. Quantoren beibehalten

Satz 3.4 *Jede prädikatenlogische Formel lässt sich in eine äquivalente Formel in pränexer Normalform transformieren.*

Von links nach rechts durchlaufen, b die Variable merke
und benenne \rightarrow stehen bleiben

Skolemisierung

Ziel = Quantoren entfernen

Klammeraufgabe!
Skolemisierung durchführen

$$\forall x_1 \forall x_2 \exists y_1 \forall x_3 \exists y_2 p(f(x_1), x_2, y_1) \vee q(y_1, x_3, y_2).$$

Skolemfunktion
(

$$\forall x_1 \forall x_2 \forall x_3 \exists y_2 p(f(x_1), x_2, g(x_1, x_2)) \vee q(g(x_1, x_2), x_3, y_2)$$

$$\forall x_1 \forall x_2 \forall x_3 p(f(x_1), x_2, g(x_1, x_2)) \vee q(g(x_1, x_2), x_3, h(x_1, x_2, x_3))$$

$$p(f(x_1), x_2, g(x_1, x_2)) \vee q(g(x_1, x_2), x_3, h(x_1, x_2, x_3)).$$

Gleichung 3.4

$$\neg gr(\varepsilon, 0) \vee \neg el(n_0(\varepsilon), \mathbb{N}) \vee \neg gr(n, n_0(\varepsilon)) \vee gr(\varepsilon, \text{abs}(\text{minus}(a(n), a))).$$

Durch den Wegfall der Variable n_0 kann die Skolemfunktion den Namen n_0 erhalten.

Skolemisierung:

y wird durch $f(x_1, \dots, x_n)$ ersetzt

$\forall x_1 \dots \forall x_n \exists y \varphi$ wird ersetzt durch $\forall x_1 \dots \forall x_n \varphi[y/f(x_1, \dots, x_n)]$

wobei die f in φ nicht vorkommen darf.

\approx eigene Variable mit y (im φ oben ist das g)

In $\exists y p(y)$, so wird y durch eine Konstante ersetzt. Wenn hier \nexists vor \exists , Skolem-Konstante

- resultierende Formel ist nicht mehr semantisch äquivalent zur Ausgangsformel
- Die Erfüllbarkeit bleibt jedoch erhalten.

Zusammenfassung der Maßnahmen, um etwas
kompliziert darzustellen

Programmschema

NORMALFORMTRANSFORMATION(*Formel*)

1. Transformation in pränexe Normalform:

Transformation in konjunktive Normalform (Satz 2.2):

Äquivalenzen eliminieren.

Implikationen eliminieren.

Wiederholte Anwendung von deMorgen-Regeln und
Distributivgesetzen.

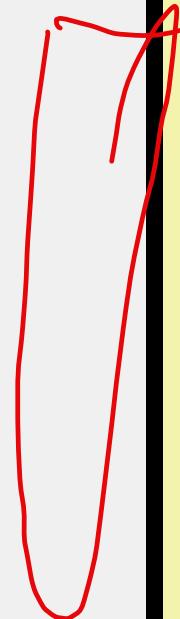
Gegebenenfalls Variablen umbenennen.

Alle Quantoren nach außen ziehen.

2. Skolemisierung:

Existenzquantifizierte Variablen durch neue Skolemfunktionen
ersetzen.

Resultierende Allquantoren löschen.



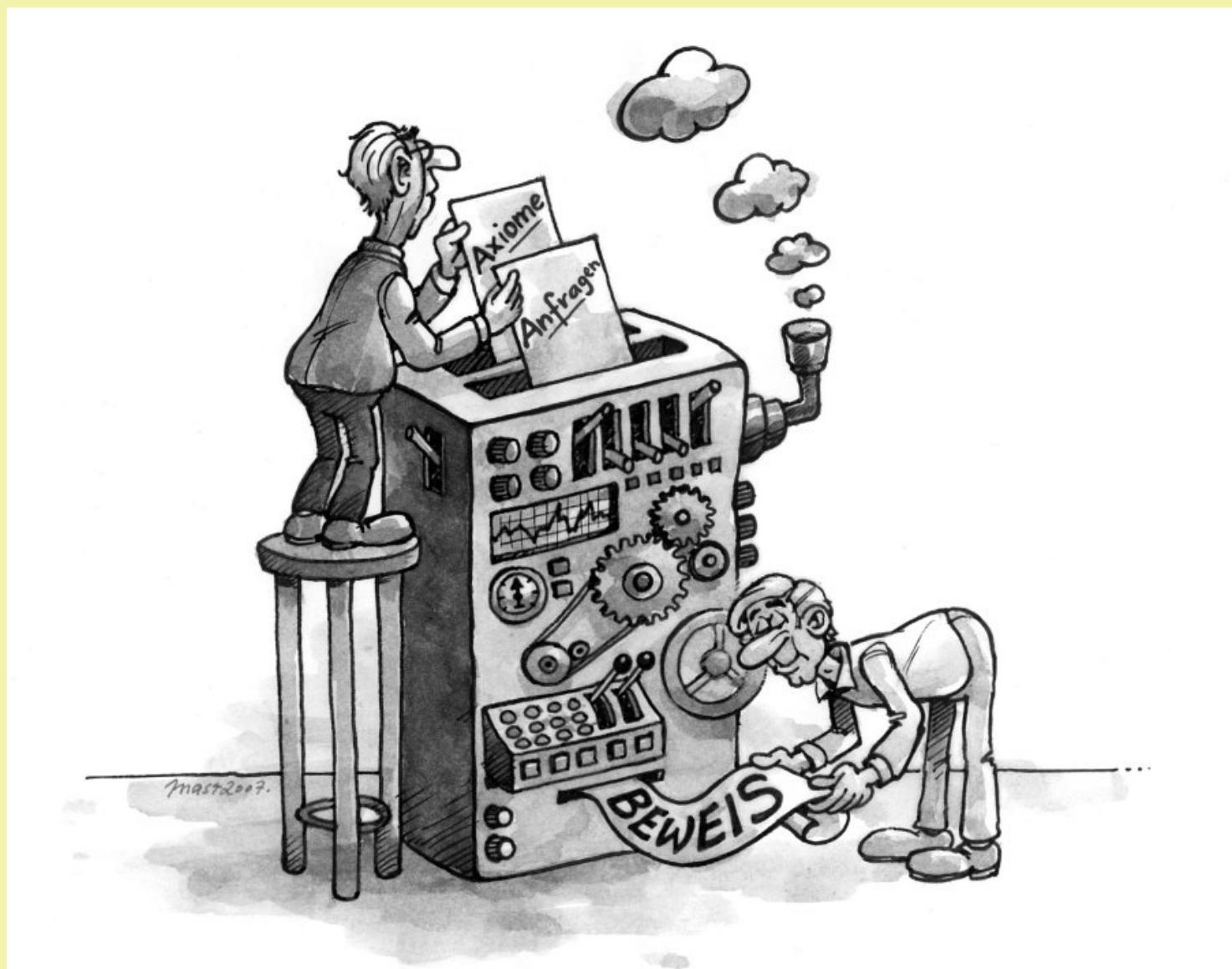
✓

Schneller & es geht nichts kaputt

Laufzeit:

1. exponentiell (naiv), 1. polynomiell Eder, 2. polynomiell

Beweiskalküle



Die universelle Logikmaschine

natürliches Schließen

Kalkül für menschliche Verarbeitung

- Gentzenkalkül
- Sequenzenkalkül
- für Anwendung durch Menschen
- intuitive Inferenzregeln
- anwendbar auf beliebige PL1-Formeln

Beispiel:

zwei einfache Inferenzregeln:

$$\frac{A, A \Rightarrow B}{B} \quad \text{benötigt 2 Formeln}$$

(Modus Ponens, MP)

$$\frac{\forall x A}{A[x/t]} \quad \text{benötigt nur 1 Formel, darf weitere [x/t] enthalten}$$

(\forall -Elimination, $\forall E$).

In Formel 4 suchen wir nach x und ersetzen sie durch t

Variable x muss ersetzt werden durch einen Grundterm t

Beweis für $\text{kind}(\text{eva}, \text{otto}, \text{anna})$:

Wissenstyp

WB :

WB :

$\forall E(2) ; x/\text{eva}, y/\text{anna}, z/\text{otto}$

MP(1, 3)

$\gamma \vdash A, 3 \vdash A = \text{S1B}$

WB :	1 $\text{kind}(\text{eva}, \text{anna}, \text{otto})$
	2 $\forall x \forall y \forall z \text{ kind}(x, y, z) \Rightarrow \text{kind}(x, z, y)$
	3 $\text{kind}(\text{eva}, \text{anna}, \text{otto}) \Rightarrow \text{kind}(\text{eva}, \text{otto}, \text{anna})$
	4 $\text{kind}(\text{eva}, \text{otto}, \text{anna})$
	$\Rightarrow \text{WB} \vdash \text{kind}(\text{eva}, \text{otto}, \text{anna})$

Kalkül

unvollständiger Kalkül!

Satz 3.6 (Gödel'scher Vollständigkeitssatz) Die Prädikatenlogik erster Stufe ist vollständig. Das heißt, es gibt einen Kalkül, mit dem sich jede Aussage φ , die aus einer Wissensbasis WB folgt, beweisen lässt. Wenn $WB \models \varphi$, dann gilt $WB \vdash \varphi$.

- Jede wahre PL1-Aussage ist beweisbar.
- Gilt auch die Umkehrung?
- Ist alles was wir syntaktisch herleiten können auch wahr?

Satz 3.8 (Korrektheit) Es gibt Kalküle, mit denen sich nur wahre Aussagen beweisen lassen. Das heißt, wenn $WB \vdash \varphi$ gilt, dann auch $WB \models \varphi$.

- Beweisbarkeit und semantische Folgerung sind äquivalent, sofern der Kalkül korrekt und vollständig ist.
- Kalküle des natürlichen Schließens für Automatisierung ungeeignet.

Resolution

Resolutionsbeweis für Beispiel 3.2:

Anfrage: $Q \equiv \neg kind(eva, otto, anna)$

Wissensbasis in konjunktiver Normalform:

$$WB \wedge \neg Q \equiv (kind(eva, anna, otto))_1 \wedge (\neg kind(x, y, z) \vee kind(x, z, y))_2 \wedge (\neg kind(eva, otto, anna))_3$$

konjunktive NF

Beweis:

$$(2) \ x/eva, \ y/anna, \ z/otto : (\neg kind(eva, anna, otto) \vee kind(eva, otto, anna))_4$$

$$\text{Res}(3, 4) : (\neg kind(eva, anna, otto))_5$$

$$\text{Res}(1, 5) : ()_6,$$

„(er Phryg) q.e.d. !“

Beispiel:

- jeder kennt seine eigene Mutter
- kennt Hans jemanden?

$$(kennt(x, mutter(x)))_1 \wedge (\neg kennt(hans, y))_2$$

Unifikation: $x/hans, y/mutter(hans)$

$$(kennt(hans, mutter(hans)))_1 \wedge (\neg kennt(hans, mutter(hans)))_2.$$

Definition 3.14 Zwei Literale heißen **unifizierbar**, wenn es eine Ersetzung σ für alle Variablen gibt, welche die Literale gleich macht. Solch ein σ wird **Unifikator** genannt. Ein Unifikator heißt allgemeinster Unifikator (engl. most general unifier (MGU)) wenn sich aus ihm alle anderen Unifikatoren durch Ersetzung von Variablen ergeben.

Beispiel: Die Literale $p(f(g(x)), y, z)$ und $p(u, u, f(u))$ sollen unifiziert werden.
Einige Unifikatoren:

$$\begin{array}{llll}
 \sigma_1 : & y/f(g(x)), & z/f(f(g(x))), & u/f(g(x)), \\
 \sigma_2 : & x/h(v), & y/f(g(h(v))), & z/f(f(g(h(v)))), & u/f(g(h(v))) \\
 \sigma_3 : & x/h(h(v)), & y/f(g(h(h(v)))), & z/f(f(g(h(h(v))))), & u/f(g(h(h(v)))) \\
 \sigma_4 : & x/h(a), & y/f(g(h(a))), & z/f(f(g(h(a)))), & u/f(g(h(a))) \\
 \sigma_5 : & x/a, & y/f(g(a)), & z/f(f(g(a))), & u/f(g(a))
 \end{array}$$

σ_1 ist der allgemeinste Unifikator. Die anderen Unifikatoren ergeben sich aus σ_1 durch die Ersetzungen $x/h(v)$, $x/h(h(v))$, $x/h(a)$, x/a .

- Prädikatssymbole werden wie Funktionssymbole behandelt
- d.h. das Literal wird wie ein Term behandelt.
- Argumente von Funktionen werden sequentiell abgearbeitet.
- Terme werden rekursiv über die Termstruktur unifiziert.

Komplexität:

- Die einfachsten Unifikationsalgorithmen sind in den meisten Fällen sehr schnell.
- Worst-Case: Rechenzeit wächst exponentiell mit der Termgröße.
- In d. Praxis scheitern fast alle Unifikationsversuche, also Worst-Case-Komplexität meist ohne Auswirkungen.
- Schnelle Unifikationsalgorithmen auch im Worst-Case fast lineare Bibel

Allgemeine Resolutionsregel für Prädikatenlogik:

Definition 3.16 Resolutionsregel für zwei Klauseln in konjunktiver Normalform

$$\frac{(A_1 \vee \dots \vee A_m \vee B), \quad (\neg B' \vee C_1 \vee \dots \vee C_n) \quad \sigma(B) = \sigma(B')}{(\sigma(A_1) \vee \dots \vee \sigma(A_m) \vee \sigma(C_1) \vee \dots \vee \sigma(C_n))}, \quad (3.6)$$

wobei σ der MGU von B und B' ist.

Satz 3.10 Die Resolutionsregel ist korrekt, das heißt, die Resolvente folgt semantisch aus den beiden Vaterklauseln.

Beispiel: Russelsche Antinomie:

“Es gibt einen Barbier der alle Menschen rasiert, die sich nicht selbst rasieren.”

In PL1 formalisiert:

$$\forall x \text{ rasiert}(\text{barbier}, x) \Leftrightarrow \neg \text{rasiert}(x, x)$$

Klauselnnormalform (siehe Aufgabe ??):

$$(\neg \text{rasiert}(\text{barbier}, x) \vee \neg \text{rasiert}(x, x))_1 \wedge (\text{rasiert}(\text{barbier}, x) \vee \text{rasiert}(x, x))_2. \quad (3.7)$$

- keinen Widerspruch ableitbar!
- also: Resolution unvollständig!

Definition 3.18 Die *Faktorisierung* einer Klausel erfolgt durch

$$\frac{(A_1 \vee A_2 \vee \dots \vee A_n) \quad \sigma(A_1) = \sigma(A_2)}{(\sigma(A_2) \vee \dots \vee \sigma(A_n))},$$

wobei σ der MGU von A_1 und A_2 ist.

Nun lässt sich aus Gleichung 3.7 ein Widerspruch ableiten

$\text{Fak}(1, \sigma : x/\text{barbier}) : (\neg \text{rasiert}(\text{barbier}, \text{barbier}))_3$

$\text{Fak}(2, \sigma : x/\text{barbier}) : (\text{rasiert}(\text{barbier}, \text{barbier}))_4$

$\text{Res}(3, 4) : ()_5.$

und es gilt allgemein

Satz 3.12 Die Resolutionsregel (3.6) zusammen mit der Faktorisierungsregel (3.18) ist vollständig, das heißt, durch Anwendung von Faktorisierungs- und Resolutionsschritten lässt sich aus jeder unerfüllbaren Formel in konjunktiver Normalform die leere Klausel ableiten.

kombinatorische Suchraumexplosion



Resolutionsstrategien

Suchraumreduktion durch spezielle Strategien:

Unit Resolution

- bevorzugt Resolutionsschritte, bei denen eine der beiden Klauseln aus nur einem Literal, einer sogenannten Unit-Klausel besteht.
- Vollständig
- Heuristik (keine garantierter Suchraumreduktion)

Set of Support Strategie.

Set of Support (SOS) $\subset WB \wedge \neg Q$

- Resolution nur zwischen Klauseln aus SOS und dem Komplement.
- Resolvente wird zum SOS hinzugefügt.
- garantierte Suchraumreduktion
- unvollständig.
- vollständig, wenn $WB \wedge \neg Q \setminus SOS$ erfüllbar
- Oft wird als initiales SOS die negierte Anfrage $\neg Q$ verwendet.

Input Resolution

- an jedem Resolutionsschritt muss eine Klausel aus der Eingabemenge $WB \wedge \neg Q$ beteiligt sein.
- garantierte Suchraumreduktion
- unvollständig.

Pure Literal Regel

- alle Klauseln werden gelöscht, die Literale enthalten, zu denen es kein komplementäres Literal in anderen Klauseln gibt.
- garantierte Suchraumreduktion
- vollständig.
- wird von praktisch allen Resolutionsbeweisern verwendet

Subsumption

- Stellen die Literale einer Klausel K_1 eine Teilmenge der Literale der Klausel K_2 dar, so kann Klausel K_2 gelöscht werden.
- Zum Beispiel ist $(\text{regnet}(\text{heute}) \Rightarrow \text{Straße_nass}(\text{heute}))$ überflüssig, wenn schon $\text{Straße_nass}(\text{heute})$ gilt.
- garantierte Suchraumreduktion
- vollständig.
- wird von praktisch allen Resolutionsbeweisern verwendet

Gleichheit

Gleichheitsaxiome:

$$\forall x \ x = x$$

$$\forall x \ \forall y \ x = y \Rightarrow y = x$$

$$\forall x \ \forall y \ \forall z \ x = y \wedge y = z \Rightarrow x = z$$



Lösung: spezielle Inferenzregeln für Gleichheit.

Demodulation:

Eine Gleichung $t_1 = t_2$ wird mittels Unifikation auf einen Term t wie folgt angewendet:

$$\frac{t_1 = t_2, \quad (\dots t \dots), \sigma(t_1) = \sigma(t)}{(\dots \sigma(t_2) \dots)}.$$

Paramodulation, arbeitet mit bedingten Gleichungen Bibel; Bläsius/Bürckert.

$t_1 = t_2$ erlaubt:

1. Ersetzung von t_1 durch t_2
2. Ersetzung von t_2 durch t_1



Lösung:

- Gleichungen werden oft nur in einer Richtung genutzt
⇒ **gerichtete Gleichungen**
- **Termersetzungssysteme**
- spezielle Gleichheitsbeweiser.

Automatische Theorembeweiser

Otter, 1984 Kalman

- erfolgreicher Resolutionsbeweiser (mit Gleichheitsbehandlung)
- L. Wos, W. McCune: Argonne National Laboratory, Chicago

“Currently, the main application of Otter is research in abstract algebra and formal logic. Otter and its predecessors have been used to answer many open questions in the areas of finite semigroups, ternary Boolean algebra, logic calculi, combinatory logic, group theory, lattice theory, and algebraic geometry.”

SETHEO, 1987 Letz et al.

- PROLOG-Technologie
- Warren Abstract Machine
- W. Bibel, J. Schumann, R. Letz: Technische Universität München
- PARTHEO: Implementierung auf Parallelrechnern

E, 2000 Schulz

- moderner Gleichheitsbeweiser
- S. Schulz: Technische Universität München

Homepage von E:

“E is a purely equational theorem prover for clausal logic. That means it is a program that you can stuff a mathematical specification (in clausal logic with equality) and a hypothesis into, and which will then run forever, using up all of your machines resources. Very occasionally it will find a proof for the hypothesis and tell you so ;-).”

Vampire

- Resolution mit Gleichheitsbehandlung
- A. Voronkov: University of Manchester, England
- Sieger auf der CADE-20, 2005

Isabelle Nipkow/Paulson/Wenzel

- interaktiver Beweiser für Prädikatenlogik höherer Stufe
- T. Nipkow, L. Paulson, M. Wenzel: Univ. Cambridge, Techn. Univ. München

Mathematische Beispiele

Anwendung von E Schulz auf:

In einer Halbgruppe sind links- und rechtsneutrales Element gleich

Definition 3.20 Eine Struktur (M, \cdot) bestehend aus einer Menge M mit einer zweistelligen inneren Verknüpfung “ \cdot ” heißt Halbgruppe, wenn das Assoziativgesetz

$$\forall x \forall y \forall z (x \cdot y) \cdot z = x \cdot (y \cdot z)$$

gilt. Ein Element $e \in M$ heißt linksneutral (rechtsneutral), wenn gilt $\forall x e \cdot x = x$ ($\forall x x \cdot e = x$).

Zu zeigen ist:

Satz 3.14 Besitzt eine Halbgruppe ein linksneutrales Element e_l und ein rechtsneutrales Element e_r , so gilt $e_l = e_r$.

Beweis, Variante 1: intuitives mathematisches Schließen

Offenbar gilt für alle $x \in M$

$$e_l \cdot x = x \quad (3.8)$$

und

$$x \cdot e_r = x \quad (3.9)$$

Setze in Gleichung 3.8 $x = e_r$ und in Gleichung 3.9 $x = e_l$ ergibt $e_l \cdot e_r = e_r$ und $e_l \cdot e_r = e_l$.

Also:

$$e_l = e_l \cdot e_r = e_r,$$

Beweis, Variante 2: Resolutionsbeweis manuell

$$\begin{array}{ll}
 (\neg e_l = e_r)_1 & \text{negierte Anfrage} \\
 (m(m(x, y), z) = m(x, m(y, z)))_2 & \\
 (m(e_l, x) = x)_3 & \\
 (m(x, e_r) = x)_4 &
 \end{array}$$

Gleichheitsaxiome:	$(x = x)_5$	(Reflexivität)
	$(\neg x = y \vee y = x)_6$	(Symmetrie)
	$(\neg x = y \vee \neg y = z \vee x = z)_7$	(Transitivität)
	$(\neg x = y \vee m(x, z) = m(y, z))_8$	Substitution in m
	$(\neg x = y \vee m(z, x) = m(z, y))_9$	Substitution in m,

Beweis:

$$\begin{aligned}
 \text{Res}(3, 6, x_6/m(e_l, x_3), y_6/x_3) : & \quad (x = m(e_l, x))_{10} \\
 \text{Res}(7, 10, x_7/x_{10}, y_7/m(e_l, x_{10})) : & (\neg m(e_l, x) = z \vee x = z)_{11} \\
 \text{Res}(4, 11, x_4/e_l, x_{11}/e_r, z_{11}/e_l) : & (e_r = e_l)_{12} \\
 \text{Res}(1, 12, \emptyset) : & ()
 \end{aligned}$$

Beweis, Variante 3: Resolutionsbeweis automatisch mit dem Beweiser E

Transformation in Klauselnormalsprache LOP:

$$(\neg A_1 \vee \dots \vee \neg A_m \vee B_1 \vee \dots \vee B_n) \quad \mapsto \quad B_1 ; \dots ; B_n \leftarrow A_1, \dots, A_m.$$

Eingabedatei für E:

```

<- gl(el,er).                                # Query
gl( m(m(X,Y),Z) , m(X,m(Y,Z)) ) .        # Assoziativität v. m
gl( m(el,X) , X ) .                         # linksneutrales El. v.
gl( m(X,er) , X ) .                         # rechtsneutrales El. v.
gl(X,X) .                                    # Gleichh: Reflexivität
gl(Y,X) <- gl(X,Y) .                        # Gleichh: Symmetrie
gl(X,Z) <- gl(X,Y) , gl(Y,Z) .              # Gleichh: Transitivität
gl( m(X,Z) , m(Y,Z) ) <- gl(X,Y) .          # Gleichh: Substitution
gl( m(Z,X) , m(Z,Y) ) <- gl(X,Y) .          # Gleichh: Substitution

```

Aufruf des Beweisers:

```
unixprompt> eproof halbgr1.lop
# Problem status determined, constructing proof object
# Evidence for problem status starts
  0 : [--gl(el,er)] : initial
  1 : [++gl(X1,X2),--gl(X2,X1)] : initial
  2 : [++gl(m(el,X1),X1)] : initial
  3 : [++gl(m(X1,er),X1)] : initial
  4 : [++gl(X1,X2),--gl(X1,X3),--gl(X3,X2)] : initial
  5 : [++gl(X1,m(X1,er))] : pm(3,1)
  6 : [++gl(X2,X1),--gl(X2,m(el,X1))] : pm(2,4)
  7 : [++gl(el,er)] : pm(5,6)
  8 : [] : sr(7,0)
  9 : [] : 8 : {proof}
# Evidence for problem status ends
```

Beweis, Variante 4: Resolutionsbeweis automatisch, ohne Gleichheitsaxiome

Eingabedatei:

```
<- el = er.                                % Query
m(m(X,Y),Z) = m(X,m(Y,Z)) .              % Assoziativität v. m
m(el,X) = X .                            % linksneutrales El. v. m
m(X,er) = X .                            % rechtsneutrales El. v. m
```

Aufruf des Beweisers:

```
unixprompt> eproof halbgr1a.lop
# Problem status determined, constructing proof object
# Evidence for problem status starts
  0 : [--equal(el, er)] : initial
  1 : [++equal(m(el,X1), X1)] : initial
  2 : [++equal(m(X1,er), X1)] : initial
  3 : [++equal(el, er)] : pm(2,1)
  4 : [--equal(el, el)] : rw(0,3)
  5 : [] : cn(4)
  6 : [] : 5 : {proof}
# Evidence for problem status ends
```

Anwendungen

- Vierfarbentheorem wurde 1976 mit Hilfe eines Spezialbeweisers erstmals bewiesen.
- Inferenz in Expertensystemen (heute weniger)
- automatische Programmverifikation
- z.B. Sicherheitseigenschaften von kryptographischen Protokollen [Fischer/Schumann](#)
- Software-Wiederverwendung

Beispiel: Software-Wiederverwendung

Spezifikation der Anfrage:

PRE_Q : Vorbedingungen, die **vor** der Anwendung des gesuchten Programms gelten müssen

$POST_Q$: Nachbedingungen, die **nach** der Anwendung des gesuchten Programms gelten müssen.

Aufgabe: Suche in einer Software-Datenbank nach Modulen, welche diese Anforderung erfüllen.

Software-Datenbank enthält zu jedem Modul M eine Beschreibung der Vorbedingungen PRE_M und der Nachbedingungen $POST_M$.

Es muss also gelten:

$$PRE_Q \Rightarrow PRE_M.$$

Wenn zum Beispiel ein Modul in der Datenbank nur Listen von ganzen Zahlen akzeptiert, so müssen in der Vorbedingung der Anfrage auch Listen von ganzen

Zahlen als Eingabe auftreten. Eine Zusatzbedingung, in der Anfrage, dass zum Beispiel nur gerade Zahlen auftreten, stört hier nicht.

Außerdem muss gelten:

$$POST_M \Rightarrow POST_Q$$

nach der Anwendung des Moduls müssen alle Eigenschaften, die die Anfrage fordert, erfüllt sein.

Beispiel: VDML-SL, Vienna Development Method - Specification language

Software-Datenbank: ROTATE erzeugt einen zyklischen Shift der Listenelemente.

ROTATE($l : List$) $l' : List$

pre true

post $(l = [] \Rightarrow l' = []) \wedge$
 $(l \neq [] \Rightarrow l' = (tll)^\wedge[hdl])$

Gesucht: SHUFFLE erzeugt eine beliebige Permutation (Vertauschung der Elemente) der Liste.

$\text{SHUFFLE}(x : \text{List}) \ x' : \text{List}$

pre true

post $\forall i : \text{Item} \ . \ (\exists x_1, x_2 : \text{List} \ . \ x = x_1 \hat{[i]} x_2 \Leftrightarrow \exists y_1, y_2 : \text{List} \ . \ x' = y_1 \hat{[i]} y_2)$

Zu beweisen: $(PRE_Q \Rightarrow PRE_M) \wedge (POST_M \Rightarrow POST_Q)$ folgt aus der Wissensbasis WB

$$\begin{aligned} & \forall l, l', x, x' : \text{List} \cdot (l = x \wedge l' = x' \wedge (w \Rightarrow w)) \wedge \\ & (l = x \wedge l' = x' \wedge ((l = \emptyset \Rightarrow l' = \emptyset) \wedge (l \neq \emptyset \Rightarrow l' = (t \mid l)^\wedge [hd\, l])) \\ \Rightarrow & \forall i : \text{Item} \cdot (\exists x_1, x_2 : \text{List} \cdot x = x_1^\wedge [i]^\wedge x_2 \Leftrightarrow \exists y_1, y_2 : \text{List} \cdot x' = y_1^\wedge [i]^\wedge y_2)). \end{aligned}$$

WB enthält eine Beschreibung des Datentyps **List**

Zusammenfassung

- das Beweisen mathematischer Sätze ist automatisierbar
- Automatische Beweiser sind für Verifikationsaufgaben einsetzbar
- Für Schließen im Alltag ist PL1 nicht geeignet

Weiterführende Literatur:

- sehr gute moderne Einführung in die Logik (+ Modallogik und Temporallogik):
[Dassow](#)
- Resolution und automatische Beweiser: [Bläsius/Bürckert](#), [Bibel](#), [Chang/Lee](#)
- für die KI relevante Logiken: [Görz/Rollinger/Schneeberger](#).
- Webseite zum Buch

Kapitel 4

Grenzen der Logik

Das Suchraumproblem



- automatische Beweiser schaffen 20000 Inferenzen pro Sekunde.
- Menschen schaffen nur 1 Inferenz pro Sekunde.
- Menschen lösen schwierige Probleme viel schneller.

Gründe:

- Menschen benutzen intuitive Kalküle auf höherer Ebene
- Menschen arbeiten mit Lemmas
- Menschen benutzen intuitives Metawissen (informal!)
- Menschen benutzen Heuristiken

Probleme:

- Menschen können oft intuitives Metawissen nicht verbal formulieren!
- Menschen **lernen** Heuristiken durch Erfahrung

Lösung: Anwendung von maschinellen Lernverfahren zum Lernen von Heuristiken [Ertel/Schumann/Suttner; Suttner/Ertel](#)

Beispiel: Resolution

1. Beweissteuerungsmodul bewertet die verschiedenen Alternativen für den nächsten Schritt heuristisch
2. wählt die Alternative mit der besten Bewertung
3. Bewertung der verfügbaren Klauseln aufgrund von Attributen der Klauseln wie etwa die Zahl der Literale, die Zahl der positiven Literale, die Komplexität der Terme
4. Man verwendet maschinelle Lernverfahren, um aus erfolgreichen Beweisen zu lernen [Ertel/Schumann/Suttner; Suttner/Ertel](#)
5. erfolgreiche Resolutionsschritte werden als positiv gespeichert
6. erfolglose Resolutionen werden als negativ gespeichert
7. maschinelles Lernverfahren generiert Programm zur Bewertung von Klauseln

Lösung: interaktive Systeme

- Mathematica, Maple oder MuPad
- Isabelle Nipkow/Paulson/Wenzel
- Omega Siekmann/Benzmüller
- MKM¹

¹www.mathweb.org/mathweb/demo.html

Entscheidbarkeit und Unvollständigkeit

- es gibt korrekte und vollständige Kalküle und Theorembeweiser
- jeder Satz kann in endlicher Zeit bewiesen werden!
- Was ist, wenn die Aussage nicht wahr ist?

Satz 4.2 Die Menge der allgemeingültigen Formeln der Prädikatenlogik erster Stufe ist halbentscheidbar.

Aufgabe ??

Die Aussagenlogik ist entscheidbar!

Logik höherer (zweiter) Stufe:

“**Wenn eine Prädikat $p(n)$ für n gilt, so gilt auch $p(n + 1)$** ”,

beziehungsweise

$$\forall p \ p(n) \Rightarrow p(n + 1)$$

Satz 4.4 (*Gödelscher Unvollständigkeitssatz*) Jedes Axiomensystem für die *Natürlichen Zahlen mit Addition und Multiplikation* (die Arithmetik) ist unvollständig. Das heißt, es gibt in der Arithmetik wahre Aussagen, die nicht beweisbar sind.

Der Gödelsche Beweis arbeitet mit der sogenannten Gödelisierung.

Hier wird jede arithmetische Formel eindeutig als Zahl kodiert (Gödelnummer).

Die Gödelisierung wird verwendet um die Formel

$F = \text{"Ich bin nicht beweisbar."}$

in der Arithmetik zu formulieren.

F ist wahr und nicht beweisbar.

Logiken werden unvollständig wenn die Sprache zu mächtig wird. Beispiel: PL1

Beispiel: Mengenlehre erlaubt Antinomien:

Die Menge aller Barbieren die alle rasieren, die sich nicht selbst rasieren.²

Dilemma: Sprachen, die mächtig genug sind, führen zu Widersprüchen!

²Viele weitere logische Antinomien sind zu finden in [Wiedemann](#).

Der fliegende Pinguin

1. Tweety ist ein Pinguin
2. Pinguine sind Vögel
3. Vögel können fliegen

Formalisiert in PL1 ergibt sich als Wissensbasis WB

$$\begin{aligned} & pinguin(tweety) \\ & pinguin(x) \Rightarrow vogel(x) \\ & vogel(x) \Rightarrow fliegen(x) \end{aligned}$$

$$WB \vdash fliegen(tweety)$$



Neuer Versuch: Pinguine können nicht fliegen

$$\text{pinguin}(x) \Rightarrow \neg \text{fliegen}(x)$$

$\vdash \neg \text{fliegen}(\text{tweety})$

Aber: $\vdash \text{fliegen}(\text{tweety})$

- Die Wissensbasis ist widersprüchlich.
- die Logik ist monoton:
 - neues Wissen kann altes nicht ungültig machen

Definition 4.2 Eine Logik heißt monoton, wenn für eine beliebige Wissensbasis WB und eine beliebige Formel ϕ die Menge der aus WB ableitbaren Formeln eine Teilmenge der aus $WB \cup \phi$ ableitbaren Formeln ist.

Noch ein Versuch

“(alle) Vögel außer Pinguinen können fliegen”

WB_2 :

$pinguin(tweety)$

$pinguin(x) \Rightarrow vogel(x)$

$vogel(x) \wedge \neg pinguin(x) \Rightarrow fliegen(x)$

$pinguin(x) \Rightarrow \neg fliegen(x)$

Problem gelöst!

$\vdash \neg fliegen(tweety)$

$fliegen(tweety)$ ist nicht ableitbar!

Der Rabe Abraxas

$WB_3:$

rabe(abraxas)

rabe(x) ⇒ vogel(x)

pinguin(tweety)

pinguin(x) ⇒ vogel(x)

vogel(x) ∧ ¬ pinguin(x) ⇒ fliegen(x)

pinguin(x) ⇒ ¬ fliegen(x)

Die Flugeigenschaften von Abraxas sind unklar!

WB_4 :

rabe(abraxas)

rabe(x) \Rightarrow vogel(x)

rabe(x) \Rightarrow \neg pinguin(x)

pinguin(tweety)

pinguin(x) \Rightarrow vogel(x)

vogel(x) \wedge \neg pinguin(x) \Rightarrow fliegen(x)

pinguin(x) \Rightarrow \neg fliegen(x)

Das Frame Problem

Es müssen auch alle Eigenschaften aufgeführt werden, die das Objekt nicht hat:

Frameaxiome

Planungsprobleme: Wird zum Beispiel ein blaues Haus rot angestrichen, so ist es hinterher rot.

Wissensbasis:

$$\begin{aligned} & \textit{farbe(haus,blau)} \\ & \textit{anstreichen(haus,rot)} \\ & \textit{anstreichen}(x,y) \Rightarrow \textit{farbe}(x,y) \end{aligned}$$

das Haus ist nun rot und blau.

nichtmonotone Logiken: Entfernen von Wissen (Formeln) aus der Wissensbasis möglich!

Default-Logik Eigenschaften für Objekte, die so lange gelten wie keine anderen Regeln vorhanden sind.

Im Tweety-Beispiel wäre die Regel **Vögel können fliegen** eine derartige **Default-Regel**.

Modellierung von Unsicherheit

$$P(vogel(x) \Rightarrow fliegen(x)) = 0.99$$

besser:

$$P(fliegen|vogel) = 0.99$$

zu arbeiten. Mit Hilfe von **Bayes-Netzen** lassen sich auch komplexe Anwendungen mit vielen Variablen modellieren.

andere Modellierung für

“Das Wetter ist schön”.

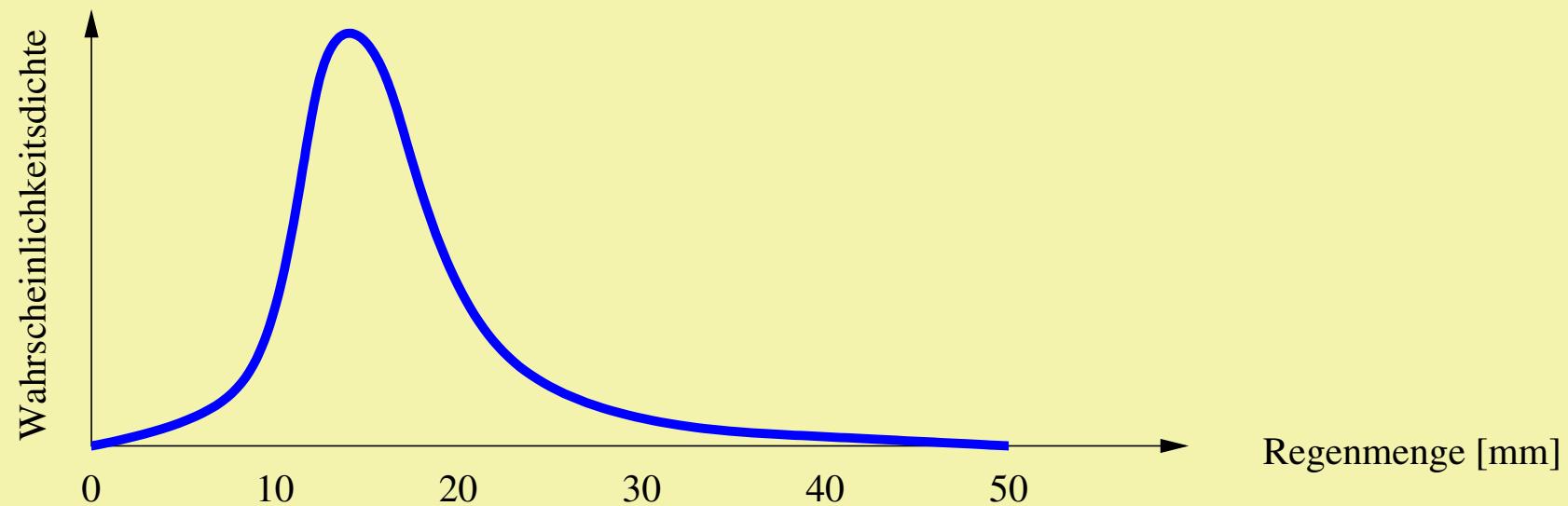
Die Variable **Wetter_ist_schön** ist stetig mit Werten in $[0, 1]$.

Wetter_ist_schön = 0.7 bedeutet dann “Das Wetter ist ziemlich schön”.

⇒ **Fuzzy-Logik**

“Mit hoher Wahrscheinlichkeit wird es ein bißchen Regen geben”

$$P(\text{Niederschlagsmenge} = X) = Y$$



Wahrscheinlichkeitsdichte der stetigen Variable Regenmenge.
sehr wahrscheinlich zwischen 10 und 20mm Regen.

Formalismus	Anzahl der Wahrheitswerte	Wahrscheinlichkeiten ausdrückbar
Aussagenlogik	2	—
Fuzzylogik	∞	—
diskrete Wahrscheinlichkeitslogik	n	ja
stetige Wahrscheinlichkeitslogik	∞	ja

Vergleich verschiedener Formalismen zur Modellierung unsicheren Wissens.

Kapitel 5

Logikprogrammierung mit PROLOG

Algorithm = Logic + Control

Anwendungen:

- KI
- Computerlinguistik

Literatur: Bratko; Clocksin/Mellish

Syntax

PL1 / Klauselnormalfom	PROLOG	Bezeichnung
$(\neg A_1 \vee \dots \vee \neg A_m \vee B)$	$B :- A_1, \dots, A_m.$	Regel
$(A_1 \wedge \dots \wedge A_m) \Rightarrow B$	$B :- A_1, \dots, A_m.$	Regel
A	$A.$	Fakt
$(\neg A_1 \vee \dots \vee \neg A_m)$	$?- A_1, \dots, A_m.$	Anfrage (Query)
$\neg(A_1 \wedge \dots \wedge A_m)$	$?- A_1, \dots, A_m.$	Anfrage (Query)

PROLOG Systeme und Implementierungen

- GNU-PROLOG Diaz
- SWI-PROLOG

PROLOG-Systeme interpretieren **Warren-abstract-machine-Code (WAM)**.

Die PROLOG-Quelldatei wird in den sogenannten WAM-Code kompiliert, welcher dann auf der WAM interpretiert wird.

Geschwindigkeit

bis zu 10 Millionen logische Inferenzen pro Sekunde (LIPS) auf 1 Gigahertz-PC

Einfache Beispiele

```
1 kind(otto,katrin,franz).  
2 kind(maria,katrin,franz).  
3 kind(eva,anna,otto).  
4 kind(hans,anna,otto).  
5 kind(isolde,anna,otto).  
6 kind(klaus,maria,ottob).  
7  
8 kind(X,Z,Y) :- kind(X,Y,Z).  
9  
10 nachkomme(X,Y) :- kind(X,Y,Z).  
11 nachkomme(X,Y) :- kind(X,U,V), nachkomme(U,Y).
```

PROLOG-Programm mit Verwandtschaftsbeziehungen (Datei: verw1.pl).

PROLOG-Interpreter

laden und compilieren:

```
?- [verw1].
```

```
?- kind(eva,otto,anna).
```

Yes

```
?- nachkomme(X,Y).
```

X = otto

Y = katrin

Yes

?- nachkomme(klaus,Y).

Y = maria

Yes

?- nachkomme(klaus,katrin).

wird nicht beantwortet.

Lösung:

```
1  nachkomme(X,Y) :- kind(X,Y,Z).  
2  nachkomme(X,Y) :- kind(X,Z,Y).  
3  nachkomme(X,Y) :- kind(X,U,V), nachkomme(U,Y).
```

Nun wird aber ?- kind(eva,otto,anna). falsch beantwortet!

Lösung:

```
1  kind_fakt(otto,katrin,franz).
2  kind_fakt(maria,katrin,franz).
3  kind_fakt(eva,anna,otto).
4  kind_fakt(hans,anna,otto).
5  kind_fakt(isolde,anna,otto).
6  kind_fakt(klaus,maria,ottob).

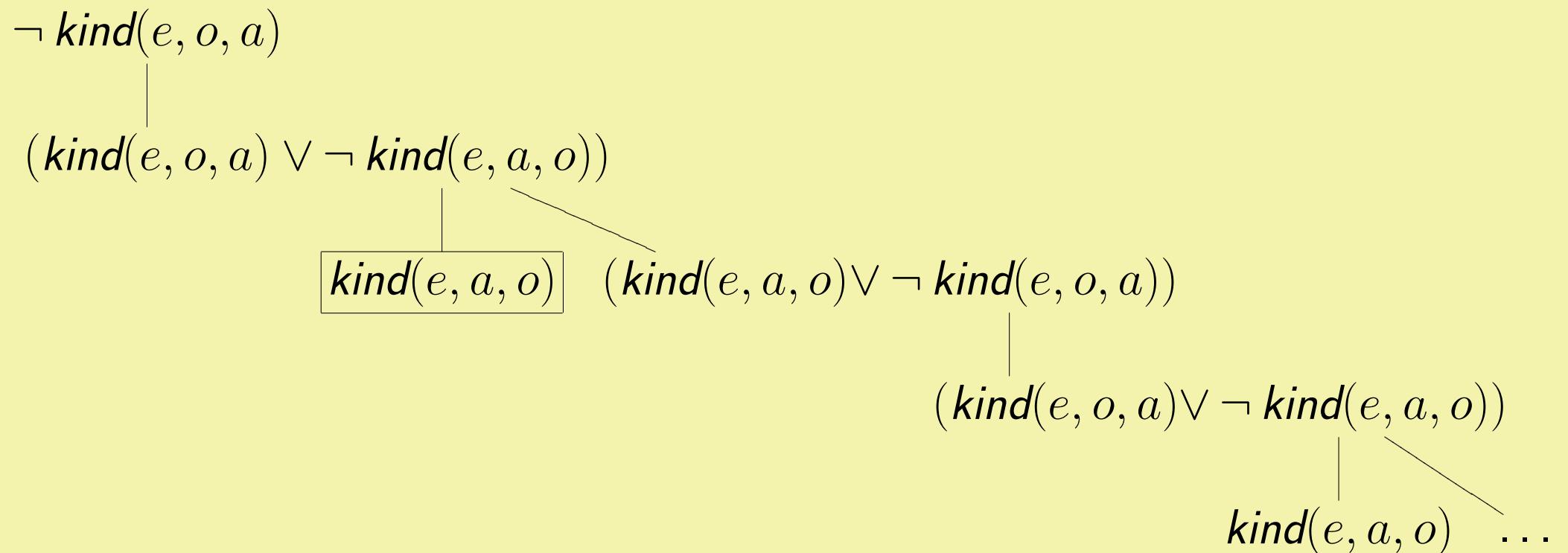
7
8  kind(X,Z,Y)  :-  kind_fakt(X,Y,Z).
9  kind(X,Z,Y)  :-  kind_fakt(X,Z,Y).

10
11 nachkomme(X,Y)  :-  kind(X,Y,Z).
12 nachkomme(X,Y)  :-  kind(X,U,V),  nachkomme(U,Y).
```

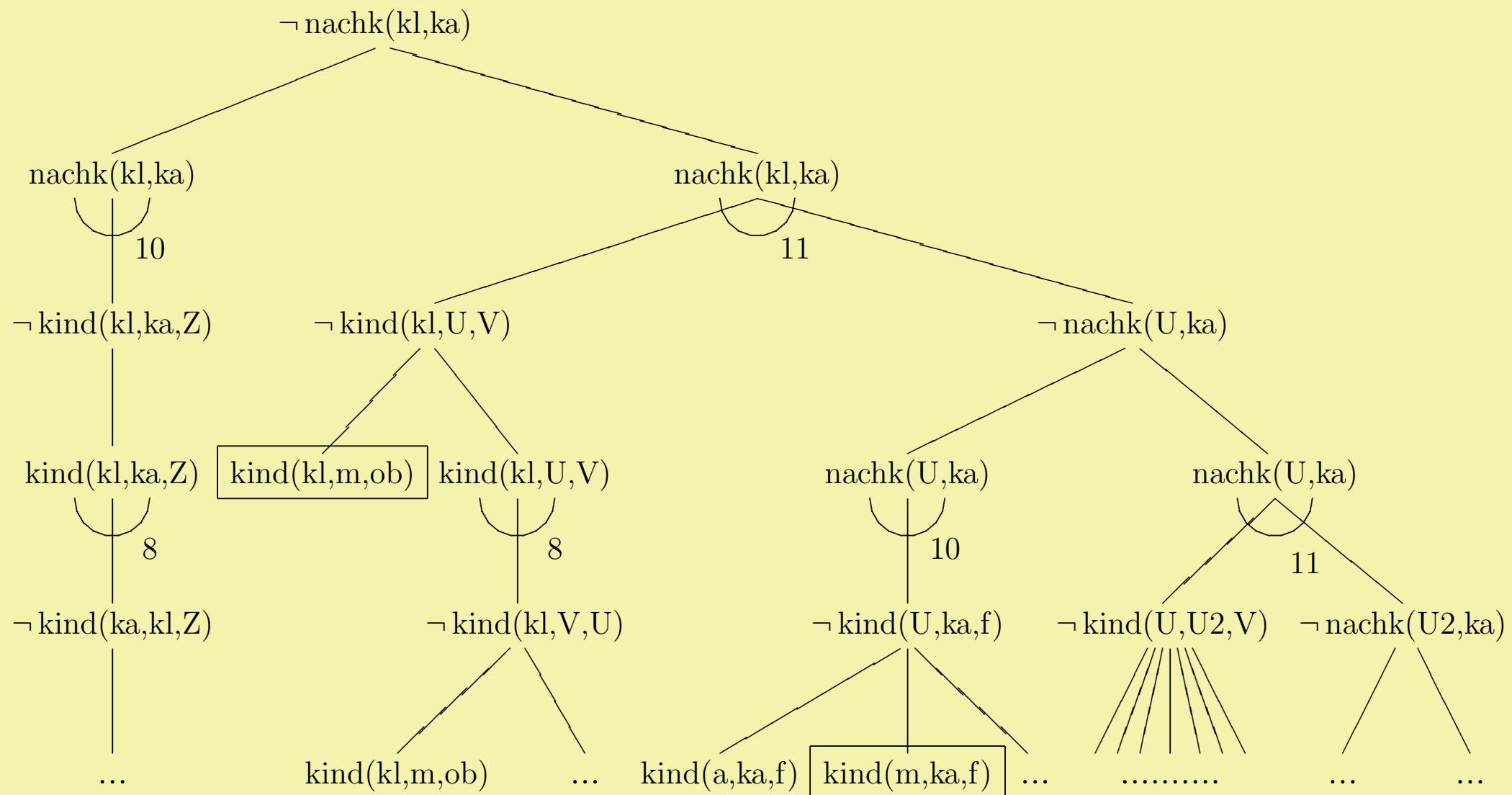
Semantik von PROLOG Programmen

- **deklarative Semantik:** logische Interpretation der Hornklauseln
- **prozedurale Semantik:** Abarbeitung der PROLOG-Programme

Suchbaum für $\text{kind}(\text{eva}, \text{otto}, \text{anna})$



Und-Oder-Baum für nachkomme(klaus, katrin)



Ablaufsteuerung und prozedurale Elemente

Vermeidung von unnötigem Backtracking durch Cut

```
1  max(X,Y,X) :- X >= Y.  
2  max(X,Y,Y) :- X < Y.
```

Anfrage:

```
?- max(2,3,Z), Z > 10.
```

Backtracking wegen $Z = 3$

Mit Cut:

```
1  max(X,Y,X) :- X >= Y, !.  
2  max(X,Y,Y).
```

Das built-in Prädikat fail

alle Kinder und deren Eltern ausgeben:

```
?- kind_fakt(X,Y,Z), write(X), write(' ist Kind von '), write(Y),  
write(' und '), write(Z), write('.'), nl, fail.
```

Ausgabe:

otto ist Kind von katrin und franz.

maria ist Kind von katrin und franz.

eva ist Kind von anna und otto.

...

No.

Was würde ohne fail am Ende ausgegeben werden?

Negation as Failure

Anfrage: ?- kind_fakt(ulla,X,Y) .

Antwort: No

Antwort ist logisch nicht korrekt!

Warum?

Der Beweiser E

würde hier korrekt antworten: "No proof found."

Die Einschränkung auf Hornklauseln

ist wichtig für die prozedurale Abarbeitung mittels SLD-Resolution.

Gegenbeispiel:

Russelsche Antinomie (Beispiel 3.5) enthält die Nicht-Hornklausel

$\text{rasiert}(\text{barbier}, X) \vee \text{rasiert}(X, X)$

Listen

[A,2,2,B,3,4,5]

[Head|Tail] trennt das erste Element (Head) vom Rest (Tail) der Liste.

WB:

```
liste([A,2,2,B,3,4,5]).
```

Dialog:

```
?- liste([H|T]).
```

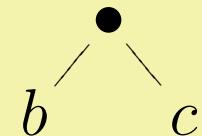
H = A

T = [2, 2, B, 3, 4, 5]

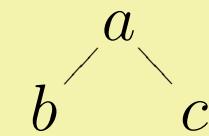
Yes

Bäume als Listen

[b , c]

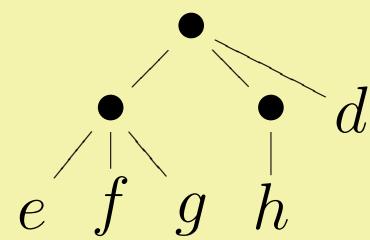


[a , b , c]



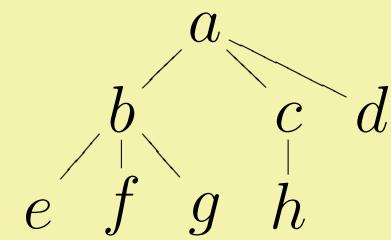
und

[[e , f , g] , [h] , d]



[a , [b , e , f , g] , [c , h] , d]

und



Das Prädikat append(X , Y , Z)

```
1  append( [] , L , L ) .  
2  append( [ X | L1 ] , L2 , [ X | L3 ] ) :- append( L1 , L2 , L3 ) .
```

- deklarative (rekursive) logische Beschreibung und gleichzeitig Abarbeitung

Beispiele:

```
?- append( [a,b,c] , [d,1,2] , Z) .
```

Z = [a, b, c, d, 1, 2]

```
?- append( X , [1,2,3] , [4,5,6,1,2,3] ) .
```

X = [4, 5, 6]

Naive Reverse

```
1  nrev([],[]).  
2  nrev([H|T],R) :- nrev(T,RT), append(RT,[H],R).
```

Besser: Liste Akkumulator

[a,b,c,d]	[]
[b,c,d]	[a]
[c,d]	[b,a]
[d]	[c,b,a]
[]	[d,c,b,a]

```
1  accrev([],A,A).  
2  accrev([H|T],A,R) :- accrev(T,[H|A],R).
```

Selbstmodifizierende Programme

```
1  :- dynamic nachkomme/2.  
2  nachkomme(X,Y) :- kind(X,Y,Z), asserta(nachkomme(X,Y)).  
3  nachkomme(X,Y) :- kind(X,U,V), nachkomme(U,Y),  
4                                asserta(nachkomme(X,Y)).
```

Die Anfrage "?- nachkomme(klaus, katrin)." führt zum Hinzufügen von:
nachkomme(klaus, katrin).
nachkomme(maria, katrin).

- Genetic Programming
- Maschinelles Lernen

Ein Planungsbeispiel

Beispiel:

Ein Bauer wollte einen Kohlkopf, eine Ziege und einen Wolf über einen Fluss bringen. Sein Boot war aber so klein, dass er entweder nur den Kohlkopf oder nur die Ziege oder nur den Wolf hinüberfahren konnte. Der Bauer dachte nach und sagte dann zu sich: "Bringe ich zuerst den Wolf ans andere Ufer, so frisst die Ziege den Kohl. Transportiere ich den Kohl als erstes, wird die Ziege vom Wolf gefressen. Was soll ich tun?"

```
1 start :- aktion(zust(links,links,links,links),  
2                   zust(rechts,rechts,rechts,rechts)).  
3  
4 aktion(Start,Ziel):-  
5     plan(Start,Ziel,[Start],Pfad),  
6     nl,write('Loesung:'),nl,  
7     write_path(Pfad).  
8  
9 plan(Start,Ziel,Besucht,Pfad):-  
10    gehe(Start,Next),  
11    sicher(Next),  
12    \+ member(Next,Besucht),          % not(member(...))  
13    plan(Next,Ziel,[Next|Besucht],Pfad).  
14 plan(Ziel,Ziel,Pfad,Pfad).  
15  
16 gehe(zust(X,X,Z,K),zust(Y,Y,Z,K)):-gegenueber(X,Y). % Bauer, Wolf  
17 gehe(zust(X,W,X,K),zust(Y,W,Y,K)):-gegenueber(X,Y). % Bauer, Ziege  
18 gehe(zust(X,W,Z,X),zust(Y,W,Z,Y)):-gegenueber(X,Y). % Bauer, Kohl  
19 gehe(zust(X,W,Z,K),zust(Y,W,Z,K)):-gegenueber(X,Y). % Bauer  
20  
21 gegenueber(links,rechts).  
22 gegenueber(rechts,links).  
23  
24 sicher(zust(B,W,Z,K)):- gegenueber(W,Z), gegenueber(Z,K).  
25 sicher(zust(B,B,B,K)).  
26 sicher(zust(B,W,B,B)).
```

Anfrage: "?- start." Antwort

Loesung:

Bauer und Ziege von links nach rechts

Bauer von rechts nach links

Bauer und Wolf von links nach rechts

Bauer und Ziege von rechts nach links

Bauer und Kohl von links nach rechts

Bauer von rechts nach links

Bauer und Ziege von links nach rechts

Yes

Definition von plan in Logik:

$$\forall z \ plan(z, z) \wedge \forall s \ \forall z \ \forall n [gehe(s, n) \wedge sicher(n) \wedge plan(n, z) \Rightarrow plan(s, z)]$$

Situationskalkül!

Constraint Logic Programming (CLP)

Problem mit PROLOG: Frame-Axiome, siehe Tweety-Beispiel:

pinguin(tweety) schließt rabe(tweety) nicht aus!

- **CLP** = Formulierung von Randbedingungen (engl. constraint) für Variablen
- Interpreter überwacht die Einhaltung aller Randbedingungen.
- Der Programmierer wird entlastet

Zitat Freuder:

“Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.”

Beispiel: Für die Raumaufteilung am Albert-Einstein-Gymnasium bei den mündlichen Abiturprüfungen soll der Hausmeister einen Plan erstellen. Er hat folgende Informationen: Die vier Lehrer Maier, Huber, Mueller und Schmid prüfen die Fächer Deutsch, Englisch, Mathe und Physik in den aufsteigend nummerierten Räumen 1, 2, 3 und 4. Jeder Lehrer prüft genau ein Fach in genau einem Raum. Außerdem weiß er folgendes über die Lehrer und ihre Fächer:

- Herr Maier prüft nie in Raum 4.
- Herr Müller prüft immer Deutsch
- Herr Schmid und Herr Müller prüfen nicht in benachbarten Räumen.
- Frau Huber prüft Mathematik.
- Physik wird immer in Raum Nr. 4 geprüft.
- Deutsch und Englisch werden nicht in Raum 1 geprüft

Wer prüft was in welchem Raum?

```
1 start .  
2     fd_domain([Maier, Huber, Mueller, Schmid],1,4),  
3     fd_all_different([Maier, Mueller, Huber, Schmid]),  
4  
5     fd_domain([Deutsch, Englisch, Mathe, Physik],1,4),  
6     fd_all_different([Deutsch, Englisch, Mathe, Physik]),  
7  
8     fd_labeling([Maier, Huber, Mueller, Schmid]),  
9  
10    Maier #\= 4,                                % Maier prüft nicht in Raum  
11    Mueller #= Deutsch,                         % Müller prüft Deutsch  
12    dist(Mueller,Schmid) #>= 2,                % Abstand Müller/Schmid >= 2  
13    Huber #= Mathe,                            % Huber prüft Mathematik  
14    Physik #= 4,                                % Physik in Raum 4  
15    Deutsch #\= 1,                             % Deutsch nicht in Raum 1  
16    Englisch #\= 1,                            % Englisch nicht in Raum 1  
17    nl,  
18    write([Maier, Huber, Mueller, Schmid]), nl,  
19    write([Deutsch, Englisch, Mathe, Physik]), nl.
```

Ausgabe:

[3,1,2,4]

[2,3,1,4]

Raumplan:

Raum Nr.	1	2	3	4
Lehrer	Huber	Müller	Maier	Schmid
Fach	Mathe	Deutsch	Englisch	Physik

Finite-Domain-Constraint-Solver

Übung: Einstein-Rätsel

Zusammenfassung

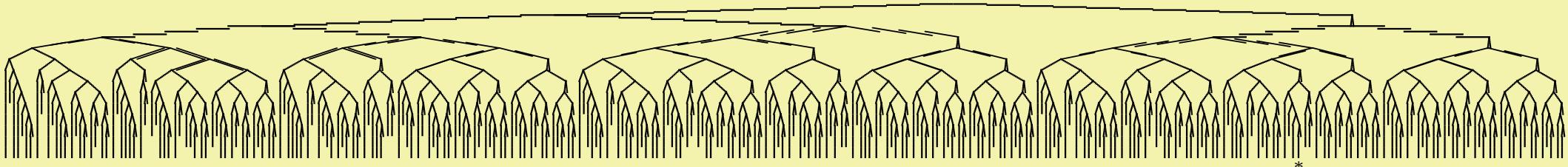
- Unifikation, Listen, deklarative Programmierung,
- relationale Sicht auf Prozeduren
- Übergabeparameter für Eingabe- und Ausgabe
- kurze Programme
- Werkzeug zum Rapid Prototyping
- CLP für Optimierungs- und Planungsaufgaben und Logikrätsel
- PROLOG in Europa, LISP in USA

Literatur: Bratko und Clocksin/Mellish, Handbücher: Wielemaker; Diaz, CLP: Bartak

Kapitel 6

Suchen, Spielen und Probleme lösen

Einführung



Suchbaum für SLD-Resolution für einfachen Beweis mit Tiefenschranke 14

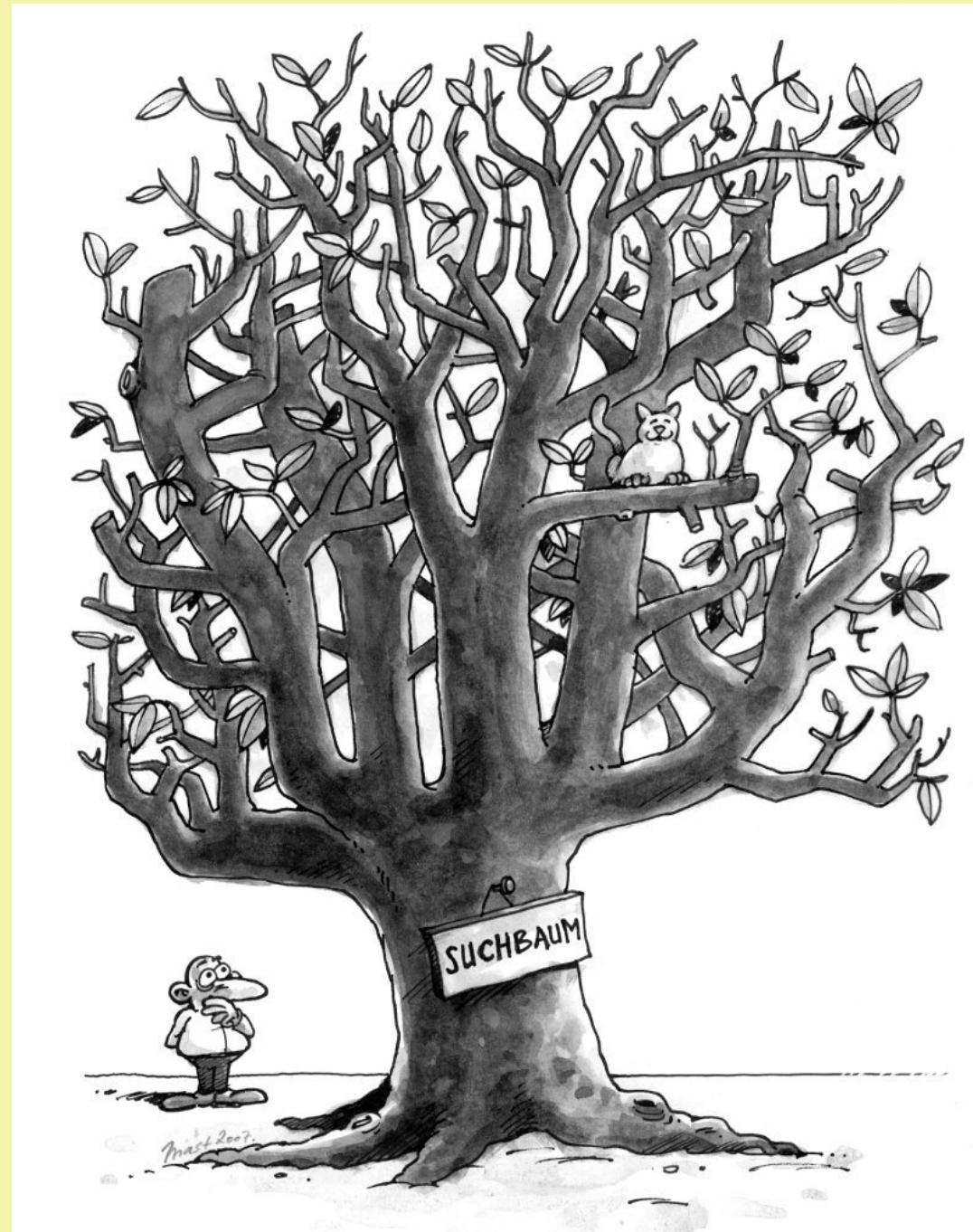
Beispiel: Schach

- Verzweigungsfaktor $b = 30$, Tiefe $d = 50$:
- $30^{50} \approx 7.2 \cdot 10^{73}$ Blattknoten.
- Anzahl Inferenzschritte = $\sum_{d=0}^{50} 30^d = \frac{1 - 30^{51}}{1 - 30} = 7.4 \cdot 10^{73}$,

- 10000 Computer
- je einer Milliarde Inferenzen pro Sekunde
- Parallelisierung ohne Verluste
- Rechenzeit:

$$\frac{7.4 \cdot 10^{73} \text{ Inferenzen}}{10000 \cdot 10^9 \text{ Inferenzen/sec}} = 7.4 \cdot 10^{60} \text{ sec} \approx 2.3 \cdot 10^{53} \text{ Jahre},$$

- 10^{43} mal Alter des Universums



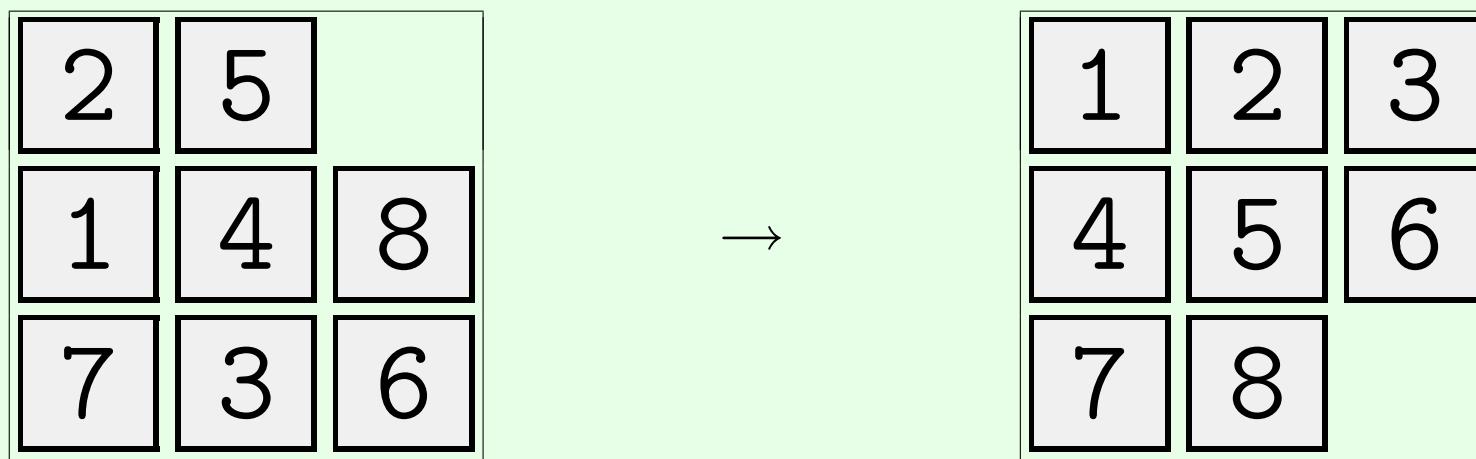
Ein stark beschnittener Suchbaum – oder: „Wo ist meine Katze geblieben?“

Fragen:

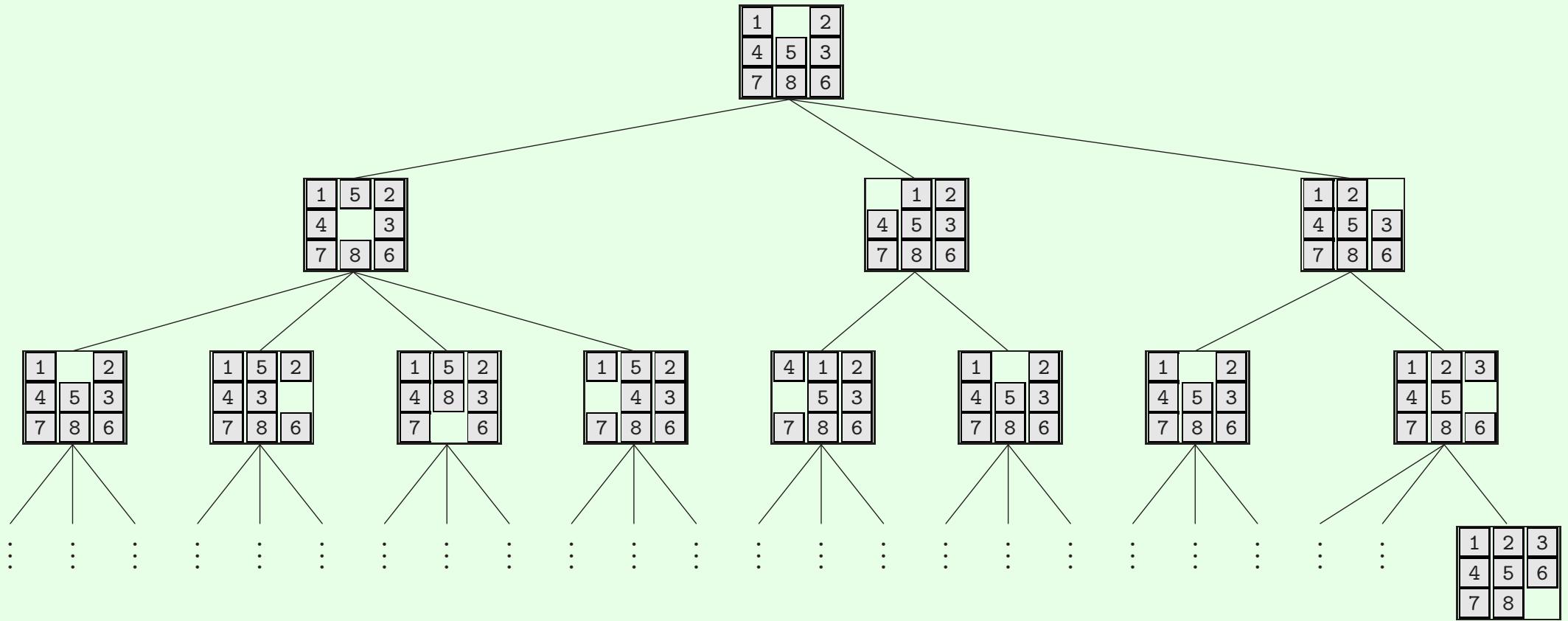
- Wie kann es dann sein, dass es gute Schachspieler – und heutzutage auch gute Schachcomputer – gibt?
- Wie kann es sein, dass Mathematiker Beweise für Sätze finden, bei denen der Suchraum noch viel größer ist?

Beispiel:

Das 8-Puzzle:

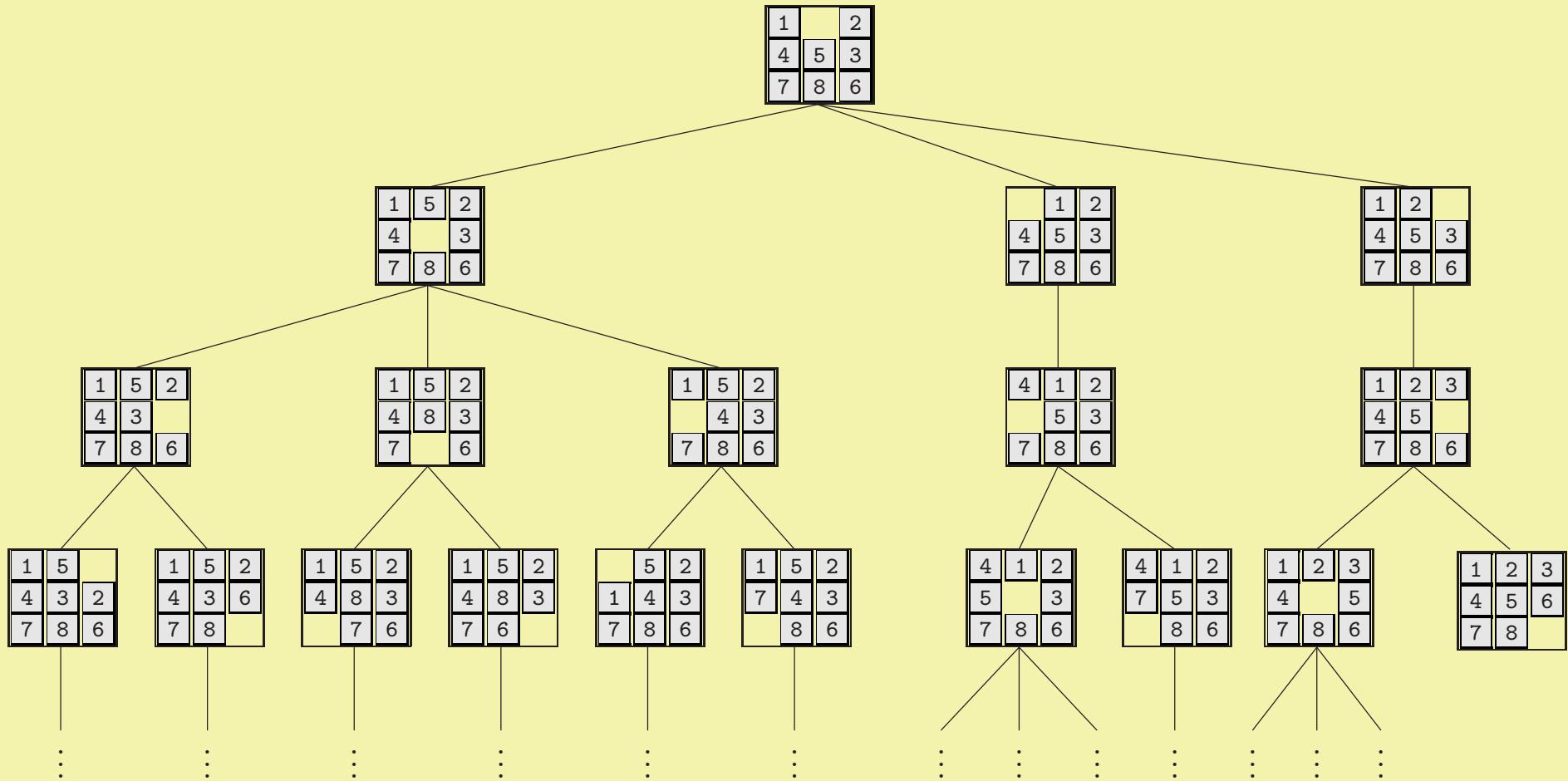


Mögliche Start- und Zielzustände des 8-Puzzle.



Definition 6.2 Der mittlere Verzweigungsfaktor eines Baumes ist der Verzweigungsfaktor, den ein Baum mit konstantem Verzweigungsfaktor, gleicher Tiefe und gleich vielen Blattknoten hätte.

mittlerer Verzweigungsfaktor = $\sqrt{8} \approx 2.83$.



mittlerer Verzweigungsfaktor $\approx 1.8^1$

¹Beim 8-Puzzle hängt der mittlere Verzweigungsfaktor vom Startzustand ab (siehe Aufgabe ??).

Definition 6.4 Ein Suchproblem wird definiert durch folgende Größen

Zustand: Beschreibung des Zustands der Welt in dem sich ein Suchagent befindet.

Startzustand: der Initialzustand in dem der Agent gestartet wird.

Zielzustand: erreicht der Agent einen Zielzustand, so terminiert er und gibt (falls gewünscht) eine Lösung aus.

Aktionen: Alle erlaubten Aktionen des Agenten.

Lösung: Der Pfad im Suchbaum vom Startzustand zum Zielzustand.

Kostenfunktion: ordnet jeder Aktion einen Kostenwert zu. Wird benötigt, um kostenoptimale Lösungen zu finden.

Zustandsraum: Menge aller Zustände.

Suchbaum: Zustände sind Knoten, Aktionen sind Kanten.

Angewandt auf das 8-Puzzle-Problem ergibt sich

Zustand: 3×3 Matrix S mit den Werten 1,2,3,4,5,6,7,8 (je einmal) und einem leeren Feld.

Startzustand: Ein beliebiger Zustand.

Zielzustand: Ein beliebiger Zustand, z.B. der in Abbildung 6.1 rechts angegebene Zustand.

Aktionen: Bewegung des leeren Feldes S_{ij} nach links (falls $j \neq 1$), rechts (falls $j \neq 3$), oben (falls $i \neq 1$), unten (falls $i \neq 3$).

Kostenfunktion: Die konstante Funktion 1, da alle Aktionen gleich aufwändig sind.

Zustandsraum: Der Zustandsraum zerfällt in Bereiche, die gegenseitig nicht erreichbar sind (Aufgabe ??). Daher gibt es nicht lösbare 8-Puzzle-Probleme.

Zur Analyse der Suchalgorithmen werden noch folgende Begriffe benötigt:

Definition 6.6

- Die Zahl der Nachfolgeszustände eines Zustands s wird als **Verzweigungsfaktor** (engl. branching faktor) $b(s)$ bezeichnet, beziehungsweise mit b , falls der Verzweigungsfaktor konstant ist.
- Der **effektive Verzweigungsfaktor** eines Baumes der Tiefe d mit insgesamt n Knoten wird definiert als der Verzweigungsfaktor, den ein Baum mit konstantem Verzweigungsfaktor, gleicher Tiefe und gleicher Knotenzahl n hätte (siehe Aufgabe ??).
- Ein Suchalgorithmus heißt **vollständig**, wenn er für jedes lösbare Problem eine Lösung findet. Terminiert ein vollständiger Suchalgorithmus ohne eine Lösung zu finden, so ist das Problem also nicht lösbar.

Auflösen der Gleichung

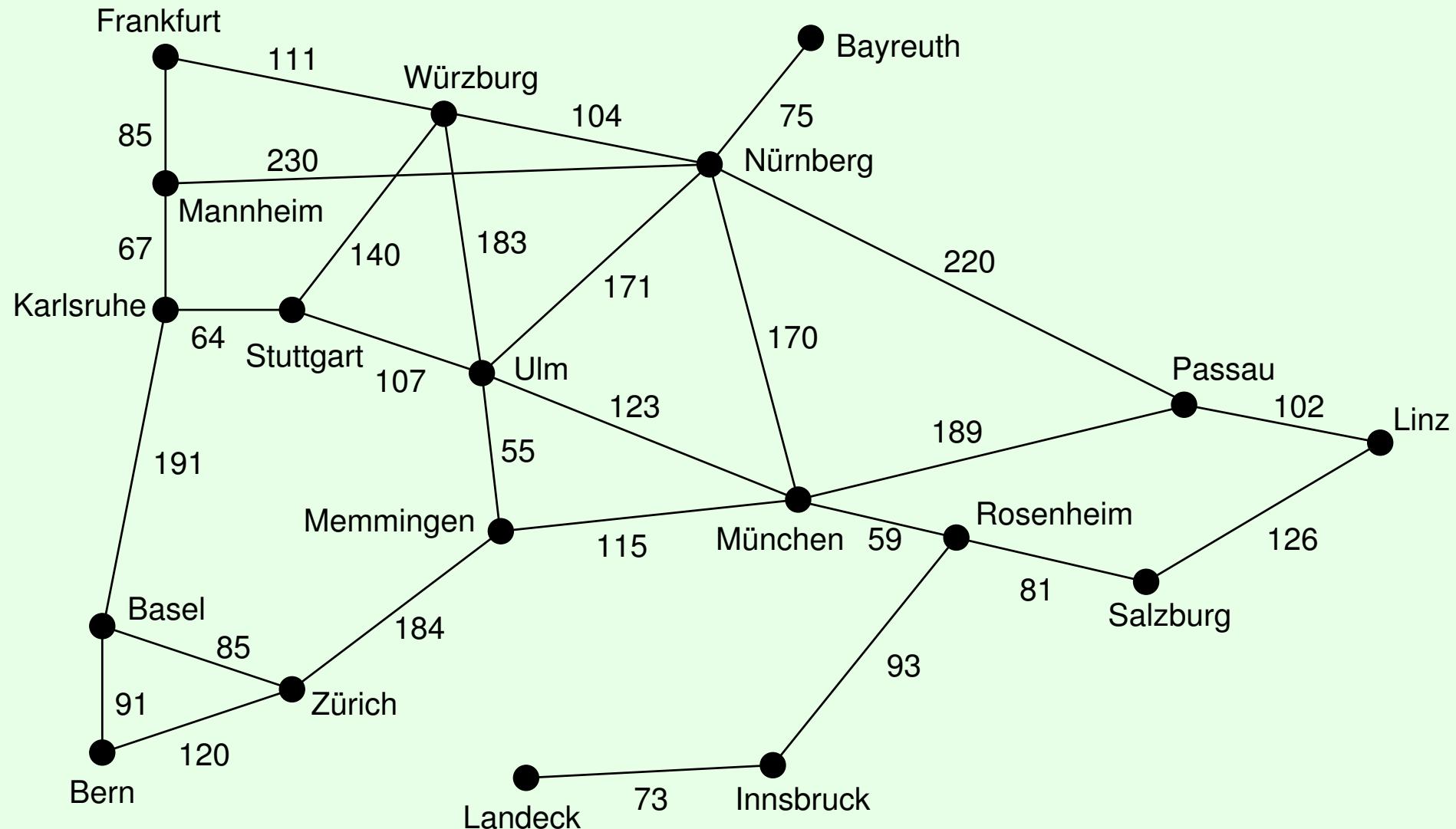
$$n = \sum_{i=0}^d b^i = \frac{b^{d+1} - 1}{b - 1}$$

nach b ergibt effektiven Verzweigungsfaktor

Satz 6.2 Bei stark verzweigenden endlichen Suchbäumen mit großem konstantem Verzweigungsfaktor liegen fast alle Knoten auf der letzten Ebene.

Beweis: (Aufgabe ??).

Beispiel: Kürzester Weg von Stadt A zu Stadt B



Der Süddeutschlandgraph mit Kostenfunktion.

Zustand: Eine Stadt als aktueller Ort des Reisenden.

Startzustand: Eine beliebige Stadt.

Zielzustand: Ein beliebige Stadt.

Aktionen: Reise von der aktuellen Stadt zu einer Nachbarstadt.

Kostenfunktion: Die Entfernung zwischen den Städten.

Zustandsraum: Alle Städte, d.h. Knoten im Graphen.

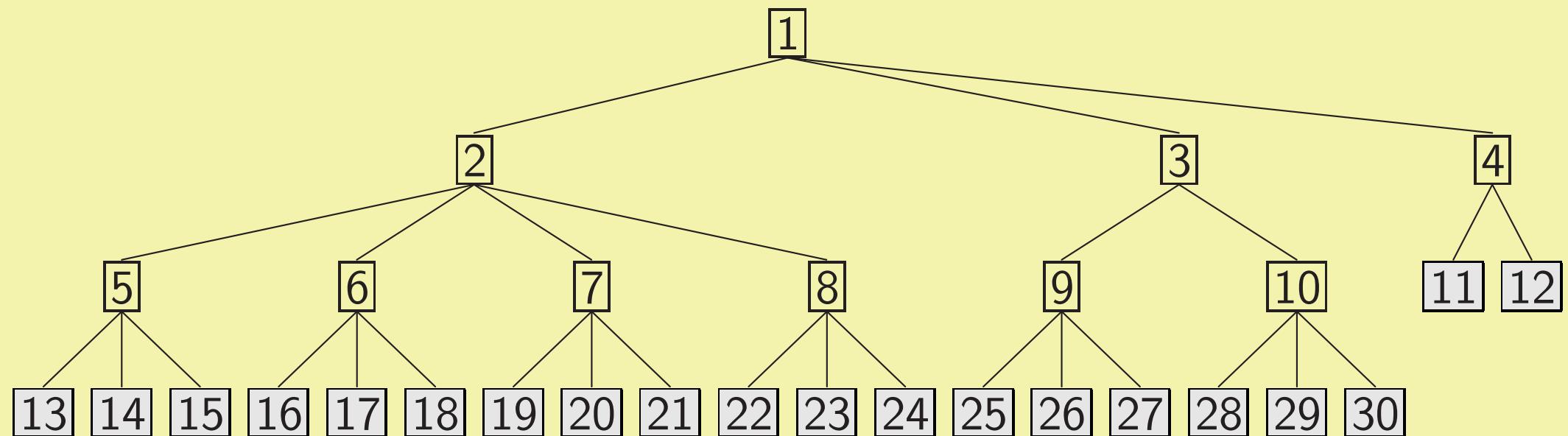
Definition 6.8 Ein Suchalgorithmus heißt **optimal**, wenn er, falls eine Lösung existiert, immer die Lösung mit den niedrigsten Kosten findet.

Das 8-Puzzle-Problem ist

- **deterministisch:** jede Aktion führt in einen eindeutig bestimmten Nachfolgezustand.
- **beobachtbar:** der Agent weiß immer, in welchem Zustand er sich befindet.
- Man kann mit sogenannten **Offline-Algorithmen** optimale Lösungen finden.
- Sonst: **Lernen durch Verstärkung**

Uninformierte Suche

Breitensuche



BREITENSUCHE(Knotenliste, Ziel)

NeueKnoten = \emptyset

For all Knoten \in Knotenliste

If Ziel_erreicht(Knoten)

Return("Lösung gefunden", Knoten)

 NeueKnoten = **Append**(NeueKnoten, Nachfolger(Knoten))

If NeueKnoten $\neq \emptyset$

Return(BREITENSUCHE(NeueKnoten, Ziel))

Else

Return("keine Lösung")

Der Algorithmus für die Breitensuche.

- Algorithmus ist generisch.
- anwendungsspezifische Funktionen "Ziel_erreicht" und "Nachfolger"

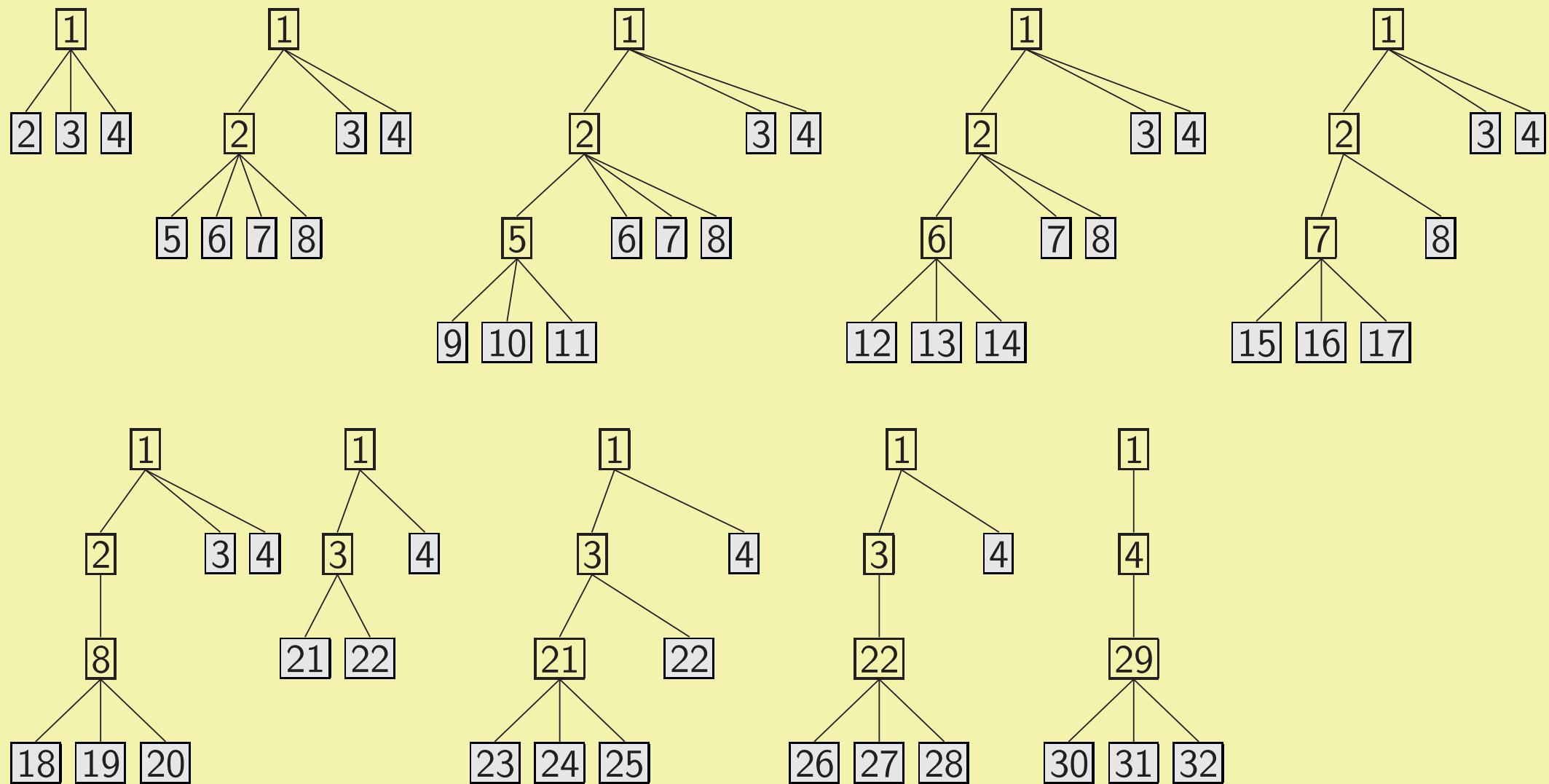
Analyse

- vollständig.
- optimal, wenn die Kosten aller Aktionen gleich sind (siehe Übung).
- Rechenzeit =

$$c \cdot \sum_{i=0}^d b^i = \frac{b^{d+1} - 1}{b - 1} = O(b^d).$$

- Speicherplatzbedarf = $O(b^d)$.
- **Uniform Cost Search:** aus aufsteigend sortierter Liste der Knoten wird immer der mit niedrigsten Kosten expandiert. Immer optimal!

Tiefensuche



TIEFENSUCHE(Knoten, Ziel)

If ZielErreicht(Knoten) **Return**("Lösung gefunden")

NeueKnoten = Nachfolger(Knoten)

While NeueKnoten ≠ ∅

Ergebnis = TIEFENSUCHE(Erster(NeueKnoten), Ziel)

If Ergebnis = "Lösung gefunden" **Return**("Lösung gefunden")

NeueKnoten = Rest(NeueKnoten)

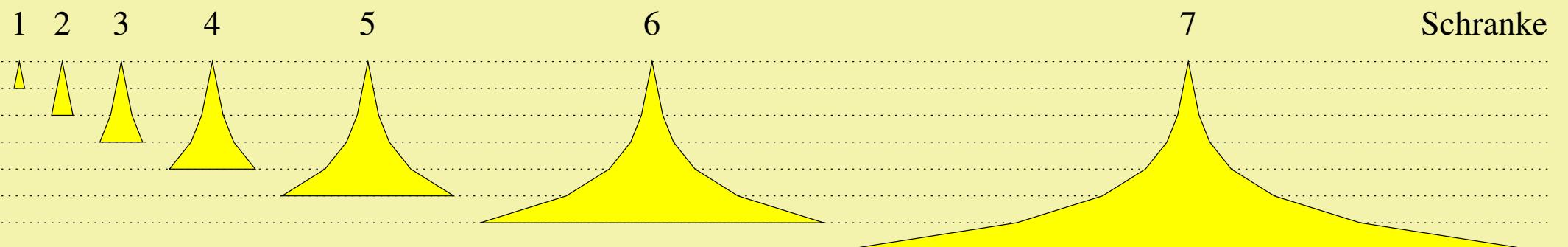
Return("keine Lösung")

Der Algorithmus für die Tiefensuche.

Analyse

- unvollständig.
- nicht optimal
- Rechenzeit = $O(b^d)$
- Speicherplatzbedarf = $O(bd)$

Iterative Deepening



ITERATIVEDEEPENING(Knoten, Ziel)

Tiefenschranke = 0

Repeat

Ergebnis = TIEFENSUCHE-B(Knoten, Ziel, 0, Tiefenschranke)

Tiefenschranke = Tiefenschranke + 1

Until Ergebnis = „Lösung gefunden“

TIEFENSUCHE-B(Knoten, Ziel, Tiefe, Schranke)

If ZielErreicht(Knoten) **Return**(„Lösung gefunden“)

NeueKnoten = Nachfolger(Knoten)

While NeueKnoten ≠ ∅ **Und** Tiefe < Schranke

Ergebnis =

TIEFENSUCHE-B(Erster(NeueKnoten), Ziel, Tiefe + 1, Schranke)

If Ergebnis = „Lösung gefunden“ **Return**(„Lösung gefunden“)

NeueKnoten = Rest(NeueKnoten)

Return(„keine Lösung“)

Analyse

- vollständig.
- optimal, wenn Kosten konstant und Inkrement = 1
- Rechenzeit = $O(b^d)$
- Speicherplatzbedarf = $\textcolor{red}{O}(bd)$

Verlust durch wiederholte Berechnungen:

$$N_b(d_{max}) = \sum_{i=0}^{d_{max}} b^i = \frac{b^{d_{max}+1} - 1}{b - 1}$$

$$\sum_{d=1}^{d_{max}-1} N_b(d) = \sum_{d=1}^{d_{max}-1} \frac{b^{d+1} - 1}{b - 1} = \frac{1}{b - 1} \left(\left(\sum_{d=1}^{d_{max}-1} b^{d+1} \right) - d_{max} + 1 \right)$$

$$\begin{aligned} &= \frac{1}{b-1} \left(\left(\sum_{d=2}^{d_{max}} b^d \right) - d_{max} + 1 \right) = \frac{1}{b-1} \left(\frac{b^{d_{max}+1} - 1}{b-1} - 1 - b \right) \\ &\approx \frac{1}{b-1} \left(\frac{b^{d_{max}+1} - 1}{b-1} \right) = \frac{1}{b-1} N_b(d_{max}) \end{aligned}$$

Für $b = 20$ etwa enthalten die ersten $d_{max} - 1$ Bäume zusammen nur etwa $\frac{1}{b-1} = 1/19$ der Knotenzahl des letzten Baumes.

Vergleich

	Breiten-suche	Uniform Cost Search	Tiefen-suche	Iterative Deepening
Vollständigkeit	ja	ja	nein	ja
Optimale Lösung	ja (*)	ja	nein	ja (*)
Rechenzeit	b^d	b^d	∞ oder b^{d_s}	b^d
Speicherplatz	b^d	b^d	bd	bd

(*) heißt, dass die Aussage nur bei konstanten Kosten gilt. d_s ist die maximale Tiefe bei endlichen Suchbaum.

Heuristische Suche

- **Heuristiken** sind Problemlösungsstrategien, die in vielen Fällen zu einer schnelleren Lösung führen als die uninformierte Suche.
- Es gibt keine Garantie!
- im Alltag sind heuristische Verfahren sehr wichtig.
- **Realzeitentscheidungen unter beschränkten Ressourcen**
- besser eine schnell gefundene gute Lösung als eine optimale, die “nie” gefunden wird.

Mathematische Modellierung

- **heuristische Bewertungsfunktion** $f(s)$ für Zustände
- **Knoten** = **Zustand** + heuristische Bewertung + ...

HEURISTISCHESUCHE(Start, Ziel)

Knotenliste = [Start]

While True

If Knotenliste = \emptyset **Return**("keine Lösung")

 Knoten = Erster(Knotenliste)

 Knotenliste = Rest(Knotenliste)

If Ziel_erreicht(Knoten) **Return**("Lösung gefunden", Knoten)

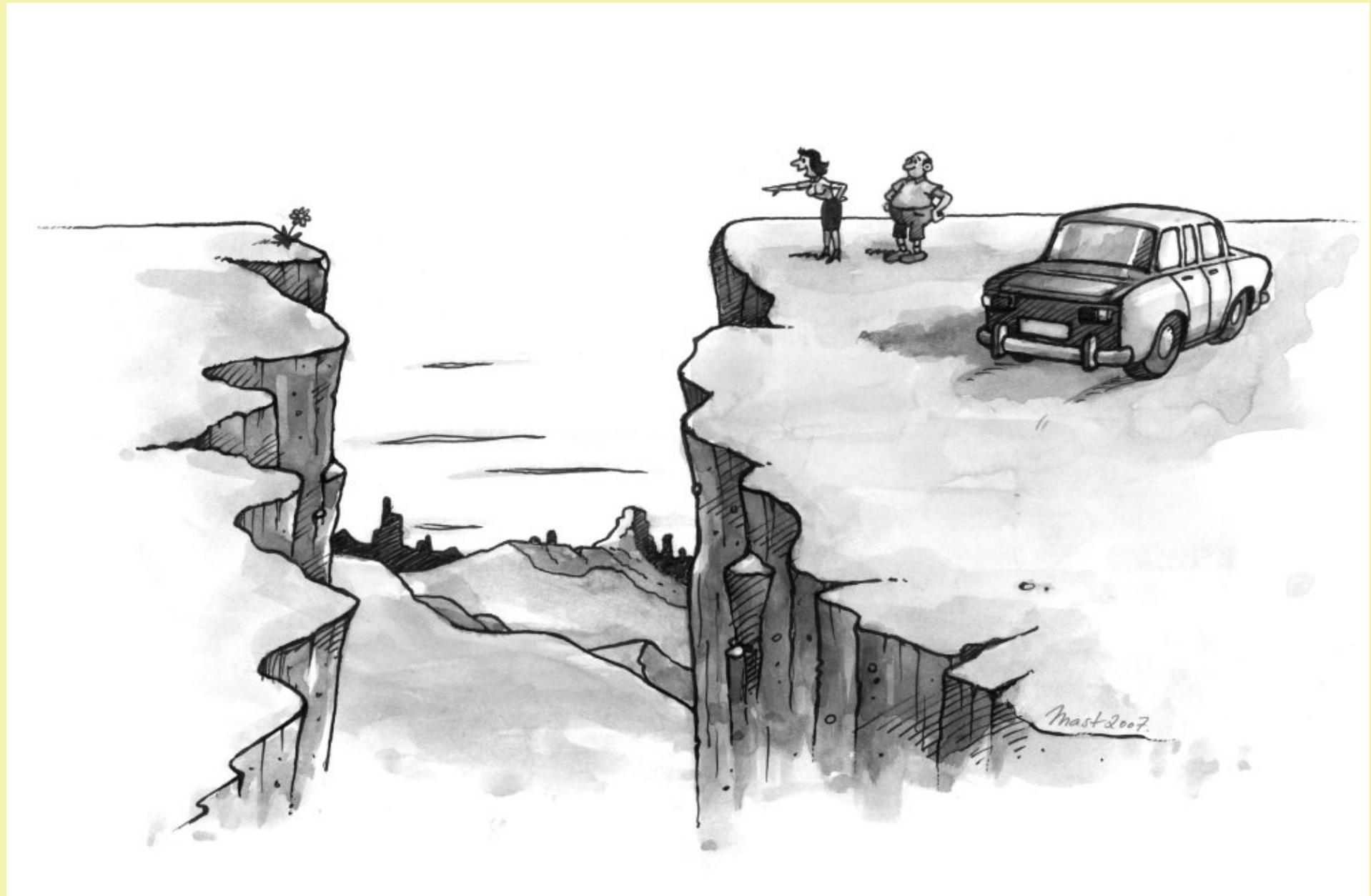
 Knotenliste = Einsortieren(Nachfolger(Knoten),Knotenliste)

Der Algorithmus für die heuristische Suche.^a

^aWährend des Einsortierens eines neuen Knotens in die Knotenliste kann es eventuell vorteilhaft sein, zu prüfen, ob der neue Knoten schon vorhanden ist und gegebenenfalls das Duplikat zu löschen.

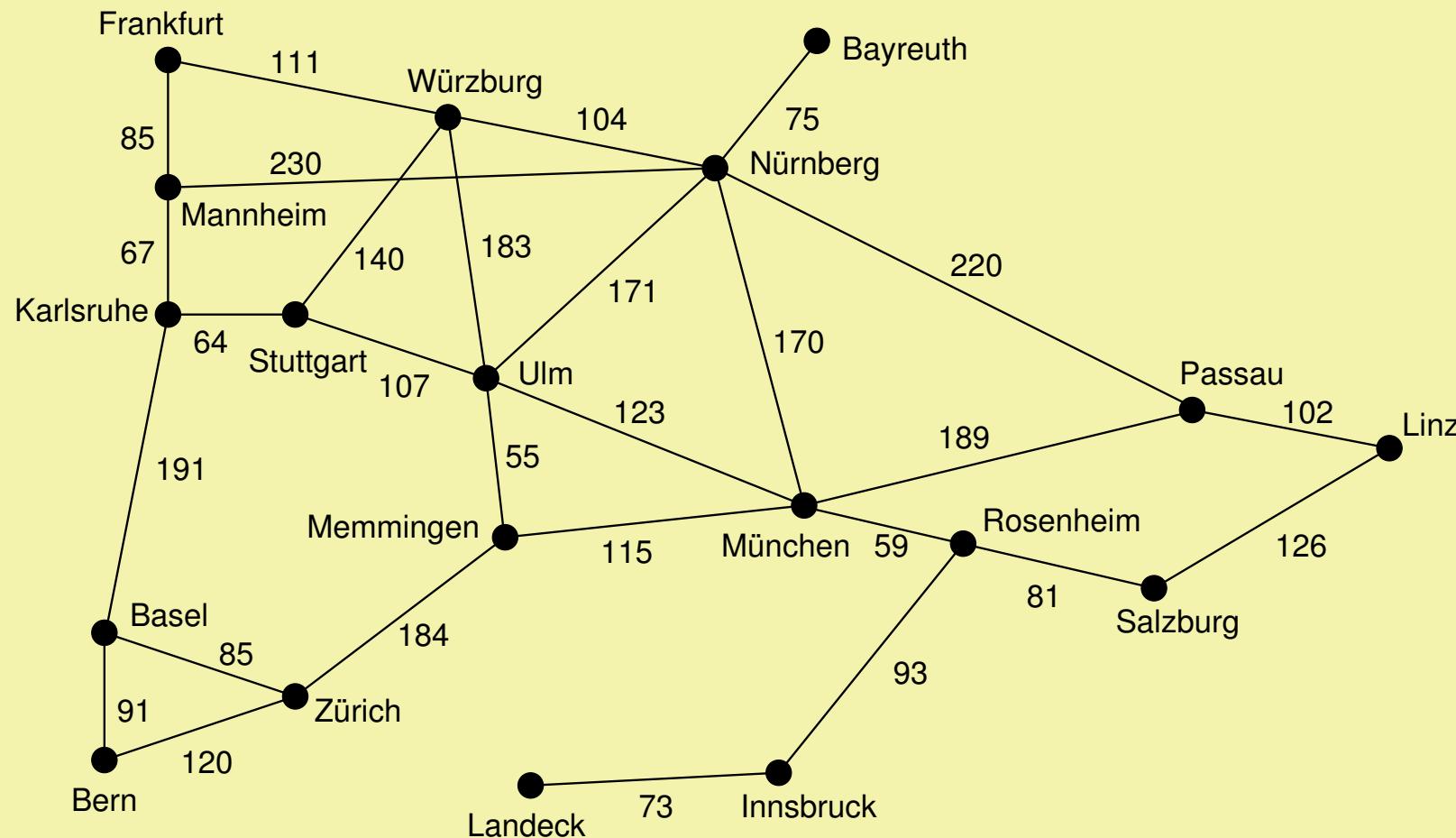
Tiefensuche und Breitensuche sind Spezialfälle der Funktion HEURISTISCHESUCHE. (Aufgabe ??)

- Die beste Heuristik wäre eine Funktion, die für jeden Knoten die tatsächlichen Kosten zum Ziel berechnet.
- Real: **Kostenschätzfunktion h**



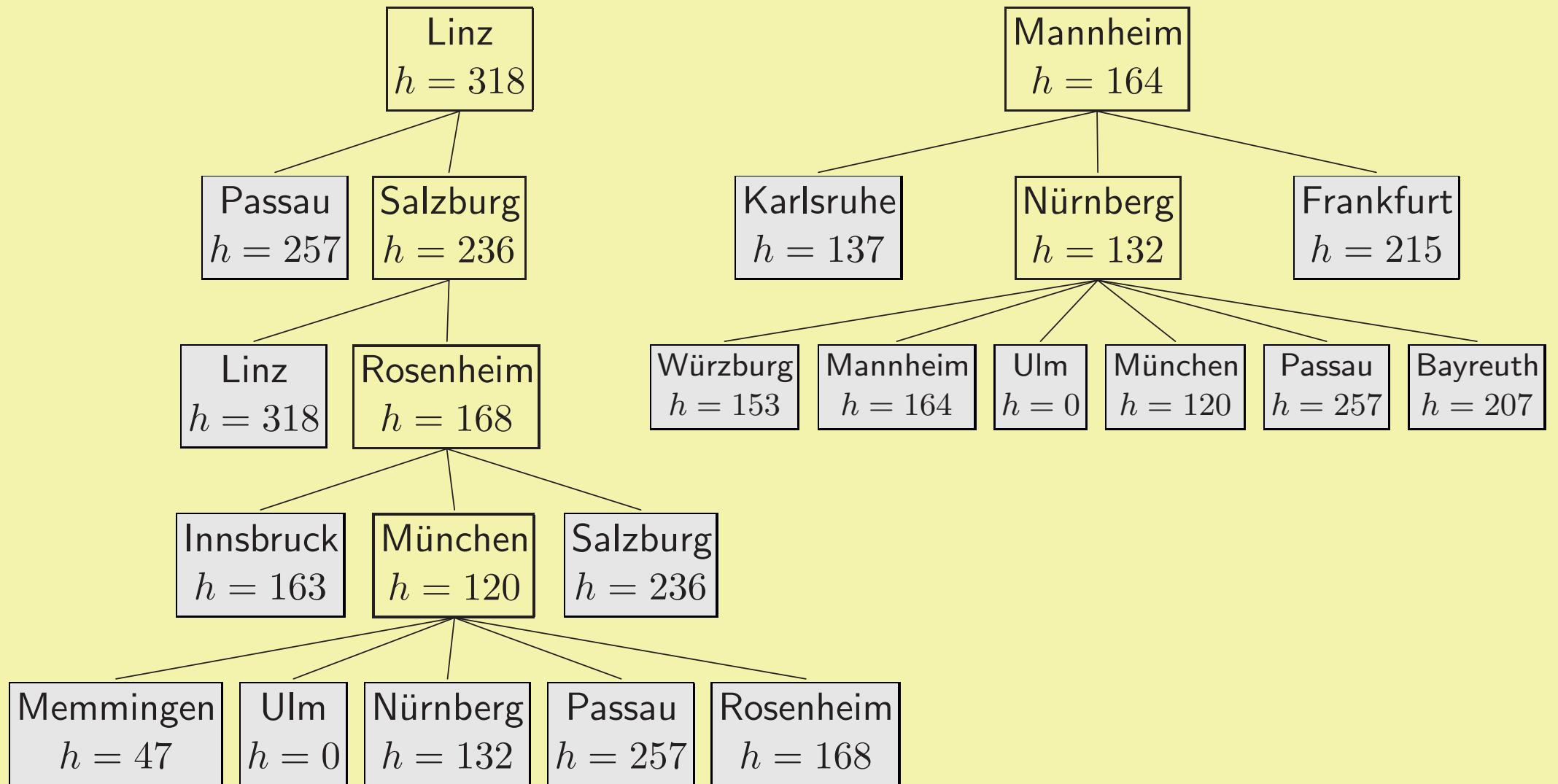
Er: „Schatz, denk mal an die Spritkosten! Ich pflück dir woanders welche.“ Sie:
„Nein, ich will die da!“

Gierige Suche



Basel	204
Bayreuth	207
Bern	247
Frankfort	215
Innsbruck	163
Karlsruhe	137
Landeck	143
Linz	318
München	120
Mannheim	164
Memmingen	47
Nürnberg	132
Passau	257
Rosenheim	168
Stuttgart	75
Salzburg	236
Würzburg	153
Zürich	157

Kostenschätzfunktion $h(s) = \text{Luftliniendistanz von Stadt } s \text{ nach Ulm.}$



- Mannheim–Nürnberg–Ulm: 401 km
- Mannheim–Karlsruhe–Stuttgart–Ulm: 323 km

A^{*}-Suche

Kostenfunktion

$g(s)$ = Summe der vom Start bis zum aktuellen Knoten angefallen Kosten,

Heuristische Schätzung

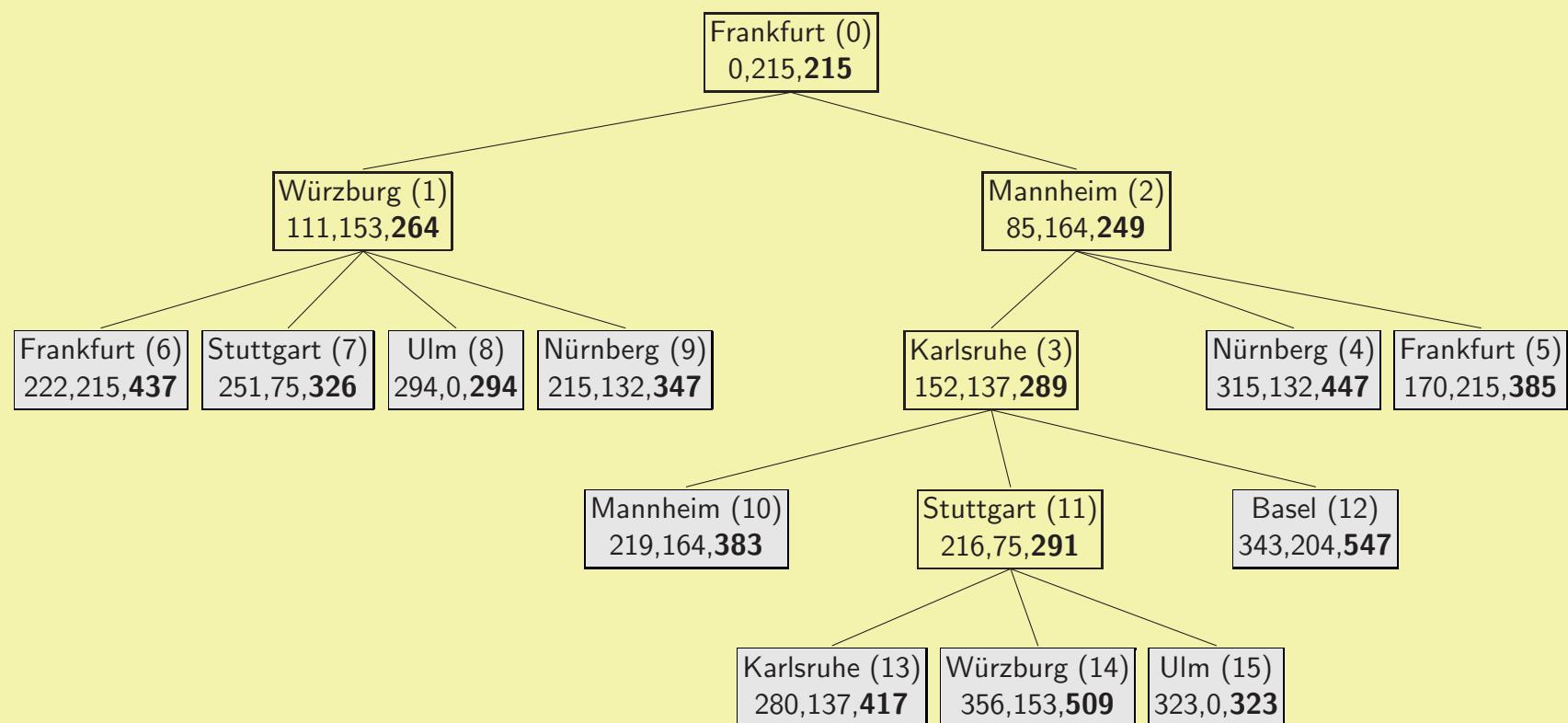
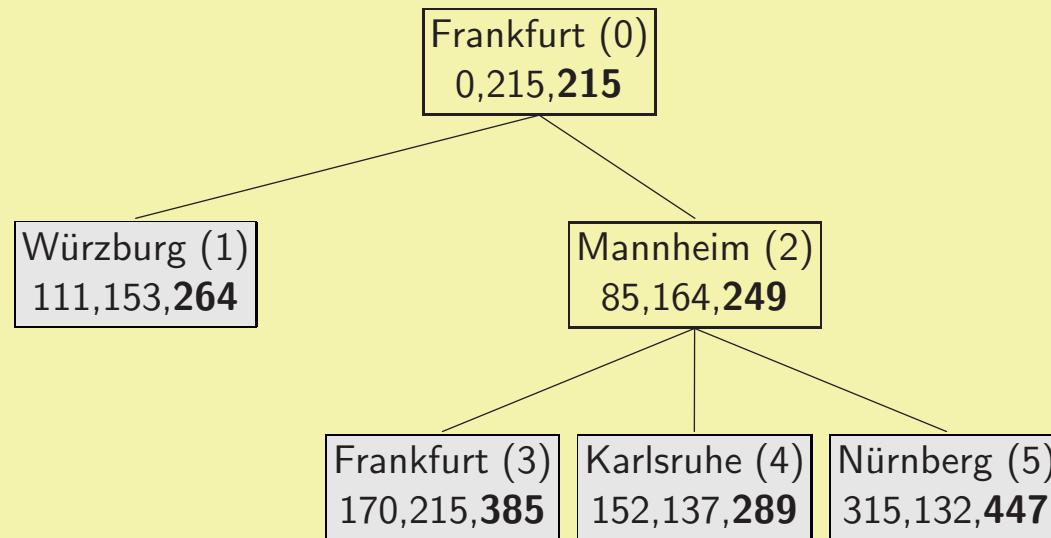
$h(s)$ = Geschätzte Kosten vom aktuellen Knoten zum Ziel

heuristische Bewertungsfunktion $f(s) = g(s) + h(s)$.

Forderung:

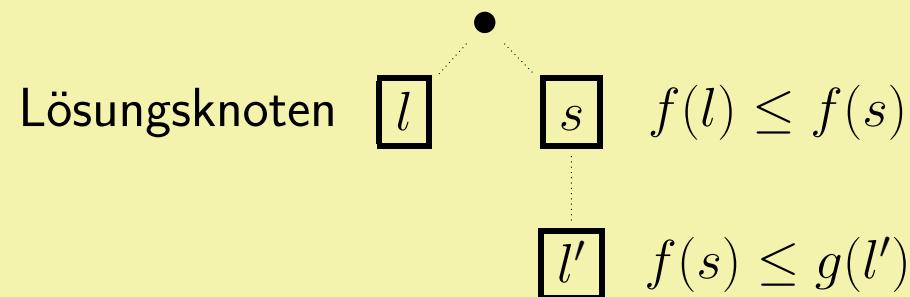
Definition 6.10 Eine heuristische Kostenschätzfunktion $h(s)$, welche die tatsächlichen Kosten vom Zustand s zum Ziel nie überschätzt, heißt **zulässig (engl. admissible)**.

A^{*}-Algorithmus = HEURISTISCHESUCHE mit Bewertungsfunktion $f(s) = g(s) + h(s)$ und zulässiger Heuristik h



Satz 6.4 Der A^* -Algorithmus ist optimal. Das heißt, er findet immer die Lösung mit den niedrigsten Gesamtkosten, wenn die Heuristik h zulässig ist.

Beweis:



Der von A^* zuerst gefundene Lösungsknoten l hat nie höhere Kosten als ein beliebiger anderer Lösungsknoten l' .

$$g(l) = g(l) + h(l) = f(l) \leq f(s) = g(s) + h(s) \leq g(l').$$

IDA^{*}-Suche

Schwächen von A^{*}:

- hoher Speicherplatzbedarf
- Liste der offenen Knoten muss sortiert sein \Rightarrow Heapsort

Lösung: Iterative Deepening

- Wie Tiefensuche, aber
- Schranke für die heuristische Bewertung $f(s)$.

Empirischer Vergleich der Suchalgorithmen

zulässige Heuristiken Für das 8-Puzzle:

- h_1 zählt die Anzahl der falschen Plättchen
- h_2 misst den **Manhattan-Abstand**

Beispiel:

Abstand zwischen

2	5	
1	4	8
7	3	6

und

1	2	3
4	5	6
7	8	

$$h_1(s) = 7, \quad h_2(s) = 1 + 1 + 1 + 1 + 2 + 0 + 3 + 1 = 10$$

h_1 und h_2 sind zulässig!

Vergleich der Suchalgorithmen

- mit Mathematica implementiert
- Mittelwerte über 132 zufällig generierte lösbare 8-Puzzle-Probleme

Vergleich

	Iterative Deepening			A [★] -Algorithmus				
Tiefe	Schritte	Zeit [sec]	Heuristik h_1		Heuristik h_2		Anz. Läufe	
			Schritte	Zeit [sec]	Schritte	Zeit [sec]		
2	20	0.003	3.0	0.0010	3.0	0.0010	10	
4	81	0.013	5.2	0.0015	5.0	0.0022	24	
6	806	0.13	10.2	0.0034	8.3	0.0039	19	
8	6455	1.0	17.3	0.0060	12.2	0.0063	14	
10	50512	7.9	48.1	0.018	22.1	0.011	15	
12	486751	75.7	162.2	0.074	56.0	0.031	12	
			IDA [★]					
14	—	—	10079.2	2.6	855.6	0.25	16	
16	—	—	69386.6	19.0	3806.5	1.3	13	
18	—	—	708780.0	161.6	53941.5	14.1	4	

effektiver Verzweigungsfaktor:

uninformierte Suche: 2.8

Heuristik h_1 : 1.5

Heuristik h_2 : 1.3

Zusammenfassung

- für uninformierte Suche ist nur Iterative Deepening praktisch einsetzbar. Warum?
- IDA^{*} ist vollständig, schnell und speichereffizient
- **gute** Heuristiken reduzieren den effektiven Verzweigungsfaktor stark
- Was bringt die Heuristik wenn das Problem unlösbar ist?

Wie findet man gute Heuristiken?

- Manuell u.a. durch Vereinfachung des Problems
- automatischen Generieren von Heuristiken mit maschinellen Lernverfahren

Spiele mit Gegner

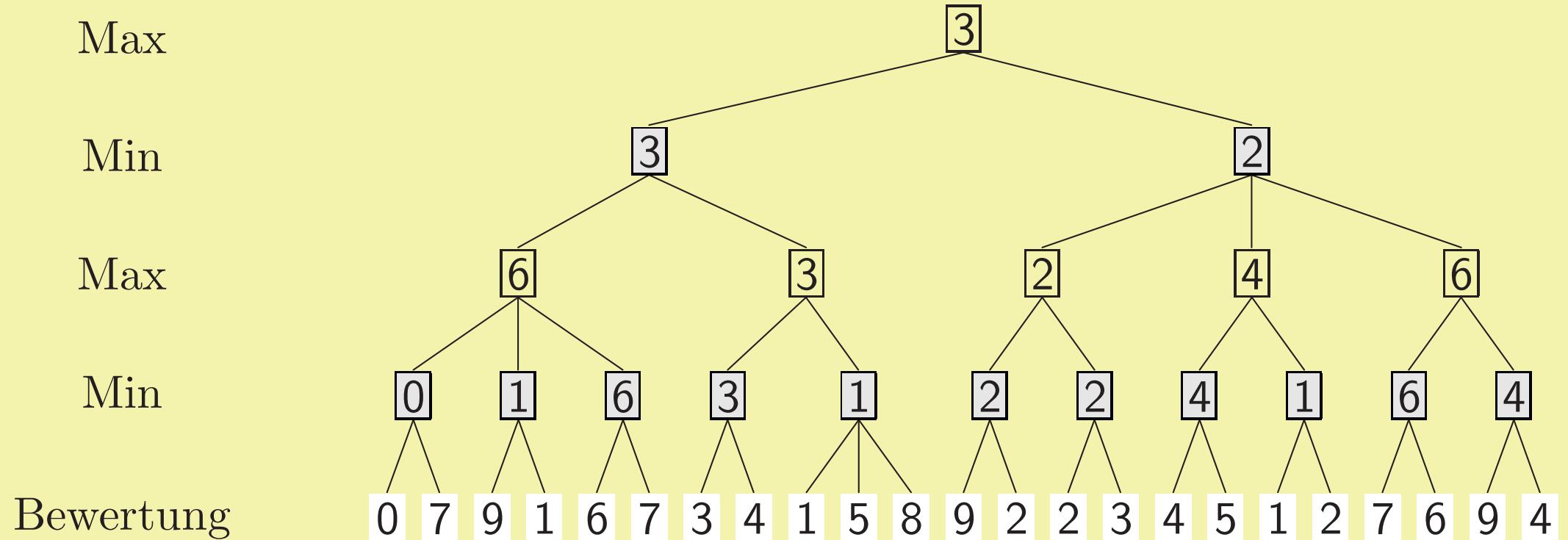
- Spiele für zwei Spieler
- Schach, Dame, Reversi, Go
- deterministisch, beobachtbar
- Kartenspiele: nur teilweise beobachtbar, warum?
- Nullsummenspiele: Gewinn + Verlust = 0

Minimax Suche

Besonderheiten bei Spielen:

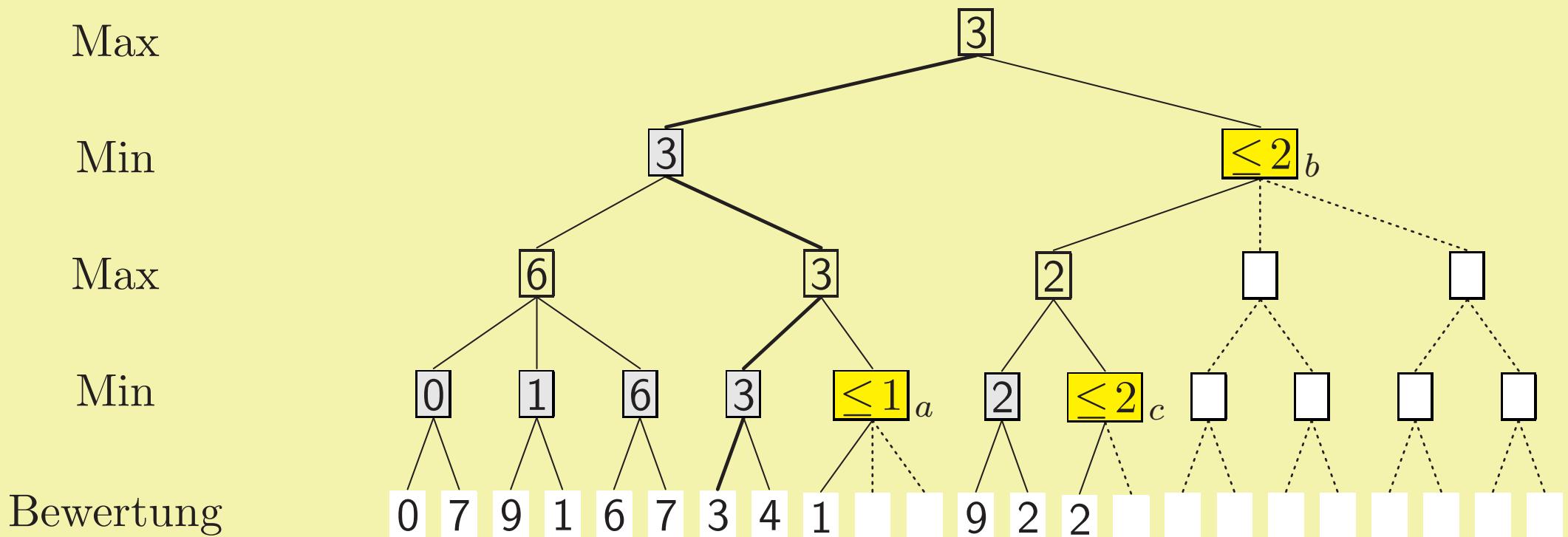
- mittlerer Verzweigungs faktor bei Schach etwa 30 bis 35
- 50 Züge pro Spieler: $30^{100} \approx 10^{148}$ Blattknoten
- Realzeitanforderungen
- beschränkte Suchtiefe
- heuristische Bewertungsfunktion b ,
- Spieler: Max, Gegner: Min
- Annahme: Gegner Min macht immer den für ihn besten Zug.
- Max maximiert Bewertung seiner Züge
- Min minimiert Bewertung seiner Züge

Minimax-Spielbaum mit Vorausschau von 4 Halbzügen.



Alpha-Beta-Pruning

Alphabeta-Spielbaum mit Vorausschau von 4 Halbzügen.



- An jedem Blattknoten wird die Bewertung berechnet.
- Für jeden Maximumknoten wird während der Suche der aktuell größte Wert der bisher traversierten Nachfolger in α gespeichert.

- Für jeden Minimumknoten wird während der Suche der aktuell kleinste Wert der bisher traversierten Nachfolger in β gespeichert.
- Ist an einem Minimumknoten k der aktuelle Wert $\beta \leq \alpha$, so kann die Suche unter k beendet werden. Hierbei ist α der größte Wert eines Maximumknotens im Pfad von der Wurzel zu k .
- Ist an einem Maximumknoten l der aktuelle Wert $\alpha \geq \beta$, so kann die Suche unter l beendet werden. Hierbei ist β der kleinste Wert eines Minimumknotens im Pfad von der Wurzel zu k .

Der in Abbildung 6.4 angegebene Algorithmus ist eine Erweiterung der Tiefensuche mit zwei sich abwechselnd aufrufenden Funktionen für die Maximum- und Minimumknoten. Er verwendet die oben definierten Werte α und β .

ALPHABETAMAX(Knoten, α , β)

If TiefenschrankeErreicht(Knoten) **Return**(Bewertung(Knoten))

NeueKnoten = Nachfolger(Knoten)

While NeueKnoten $\neq \emptyset$

α = Maximum(α , ALPHABETAMIN(Erster(NeueKnoten), α , β))

If $\alpha \geq \beta$ **Return**(β)

NeueKnoten = Rest(NeueKnoten)

Return(α)

ALPHABETAMIN(Knoten, α , β)

If TiefenschrankeErreicht(Knoten) **Return**(Bewertung(Knoten))

NeueKnoten = Nachfolger(Knoten)

While NeueKnoten $\neq \emptyset$

β = Minimum(β , ALPHABETAMAX(Erster(NeueKnoten), α , β))

If $\beta \leq \alpha$ **Return**(α)

NeueKnoten = Rest(NeueKnoten)

Return(β)

Komplexität

Rechenzeit hängt stark von der Knotenordnung ab

Worst-Case: kein Gewinn, d.h. $n_d = b^d$

Best-Case: Nachfolger von Maximumknoten absteigend sortiert, Nachfolger von Minimumknoten aufsteigend: Verzweigungsfaktor $\approx \sqrt{b}$.

$$n_d = \sqrt{b}^d = b^{d/2}$$

Schach: Verzweigungsfaktors reduziert sich von 35 auf etwa 6.

Suchhorizont wird verdoppelt.

Average-Case: Verzweigungsfaktor $\approx b^{3/4}$

Schach: Verzweigungsfaktors reduziert sich von 35 auf etwa 14.

⇒ 8 statt 6 Halbzüge vorausberechnen

Heuristische Knotenordnung: Verzweigungsfaktor ≈ 7 bis 8

Nichtdeterministische Spiele

- z.B. Würfelspiele
- Max, Würfeln, Min, Würfeln, . . . ,
- Mittelung über die Bewertungen aller Würfe

Heuristische Bewertungsfunktionen

- Liste relevanter Features oder Attribute.
- lineare Bewertungsfunktion $B(s)$

$$\begin{aligned} B(s) = & a_1 \cdot \text{Material} + a_2 \cdot \text{Bauernstruktur} + a_3 \cdot \text{Königsicherheit} \\ & + a_4 \cdot \text{Springer_im_Zentrum} + a_5 \cdot \text{Läufer_Diagonalabdeckung} + \dots \end{aligned} \tag{6.1}$$

$$\text{Material} = \text{Material}(\text{eigenes Team}) - \text{Material}(\text{Gegner})$$

$$\begin{aligned} \text{Material}(\text{Team}) = & \text{Anz. Bauern}(\text{Team}) \cdot 100 + \text{Anz. Springer}(\text{Team}) \cdot 300 \\ & + \text{Anz. Läufer}(\text{Team}) \cdot 300 + \text{Anz. Türme}(\text{Team}) \cdot 500 \\ & + \text{Anz. Damen}(\text{Team}) \cdot 900 + \dots \end{aligned}$$

Gewichtungen a_i werden vom Experten gefühlsmäßig festgelegt und optimiert

besser: Optimierung der Gewichte durch **maschinelle Lernverfahren**

Lernen von Heuristiken

- Experte legt nur die Features $f_1(s), \dots, f_n(s)$ fest.
- maschinelles Lernverfahren lernt eine möglichst optimale Bewertungsfunktion $B(f_1, \dots, f_n)$
- Bewertung am Ende, z.B.: Sieg, Niederlage, Remis
- Lernverfahren verändert Bewertungsfunktion

Problem:

- **Credit Assignment**
- keine Bewertung der einzelnen Züge!
- erst am Ende ein positives oder negatives Feedback.
- Feedback für Aktionen in der Vergangenheit?

junges Gebiet: **Lernens durch Verstärkung** (engl. reinforcement learning) (siehe Abschnitt 10).

Die weltbesten Schachcomputer arbeiten immer noch ohne Lernverfahren. Gründe:

- Lernens durch Verstärkung sehr rechenaufwändig
- Manuell erstellte Heuristiken sind schon sehr stark optimiert.

Stand der Forschung

Definition von Elaine Rich :

Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.

direkter Vergleich von Computer und Mensch in einem Spiel

1950 Claude Shannon, Konrad Zuse, John von Neumann: erste Schachprogramme

1955 Arthur Samuel: Programm, das Dame spielen und lernen konnte auf IBM 701.

Archivierte Spiele, bei denen jeder einzelne Zug von Experten bewertet war.

Programm spielt gegen sich selbst spielen.

Credit Assignment: Für jede einzelne Stellung während eines Spiels vergleicht er die Bewertung durch die Funktion $B(s)$ und die durch Alpha-Beta-Pruning berechnete Bewertung und verändert $B(s)$ entsprechend.

1961 Samuels Dame-Programm besiegt den viertbesten Damespieler der USA.

1990 Tesauro: Lernen durch Verstärkung.

lernfähiges Backgammon-Programm TD-Gammon spielte auf Weltmeister-niveau (siehe Abschnitt 10).

1997 IBM's Deep Blue besiegt mit 3.5 zu 2.5 den Schachweltmeister Gary Kasparov

Deep Blue denkt etwa 12 Halbzüge mit AlphaBeta-Pruning voraus.

2004 Hydra: Schachcomputer auf Parallelrechner

Hydra arbeitet mit 64 parallelen Xeon-Prozessoren mit je etwa 3 Ghz Rechenleistung und 1 GByte Speicher.

Software: Christian Donninger (Österreich) und Ulf Lorenz, Christopher Lutz (Deutschland).

Stellungsbewertung: FPGA-Coprozessor (Field Programmable Gate Arrays). Kann 200 Millionen Stellungen pro Sekunde bewerten.

Hydra kann etwa 18 (maximal 40) Halbzüge vorausberechnen.

oft macht Hydra erfolgreiche Züge, die Großmeister nicht nachvollziehen können.

Alpha-Beta-Suche mit relativ allgemeinen, bekannten Heuristiken und eine gute handkodierte Stellungsbewertung.

Hydra ist nicht lernfähig.

Herausforderungen heute:

- Go:
- quadratisches Brett mit 361 Feldern
- 181 weiße, 180 schwarze Stein
- mittlerer Verzweigungsfaktor: etwa 300
- nach 4 Halbzügen: $8 \cdot 10^9$ Stellungen
- klassische Spielbaum-Suchverfahren chancenlos!
- Muster auf dem Brett erkennen!
- Menschen sind den Computern noch überlegen.

Kapitel 7

Schließen mit Unsicherheit

Der fliegende Pinguin

1. Tweety ist ein Pinguin
2. Pinguine sind Vögel
3. Vögel können fliegen

Formalisiert in PL1 ergibt sich als Wissensbasis WB

$$\begin{aligned} & pinguin(tweety) \\ & pinguin(x) \Rightarrow vogel(x) \\ & vogel(x) \Rightarrow fliegen(x) \end{aligned}$$

Es lässt sich ableiten: $fliegen(tweety)$



Der fliegende Pinguin Tweety

Neuer Versuch: Pinguine können nicht fliegen

$$\text{pinguin}(x) \Rightarrow \neg \text{fliegen}(x)$$

Es lässt sich ableiten: $\neg \text{fliegen}(\text{tweety})$

Aber: Es lässt sich auch ableiten: $\text{fliegen}(\text{tweety})$

- Die Wissensbasis ist widersprüchlich.
- die Logik ist monoton:
 - neues Wissen kann altes nicht ungültig machen

Wahrscheinlichkeitslogik

Unsicherheit: *99% aller Vögel können fliegen*

Unvollständigkeit: Agent hat unvollständige Informationen über den Zustand der Welt (Realzeitentscheidungen)

Heuristische Suche

Schließen mit unsicherem und unvollständigem Wissen



„Lass uns mal in Ruhe überlegen, was wir nun machen!“

Bauchschmerzen re. u. \wedge Leukozyten > 10000 \rightarrow *Blinddarmentzündung*

Aus

Bauchschmerzen re. u. \wedge Leukozyten > 10000

ist mit Modus Ponens *Blinddarmentzündung* ableitbar.

MYCIN

- 1976, Shortliffe und Buchanan
- Certainty Factors geben Grad der Sicherheit von Fakten und Regeln an
- $A \rightarrow_{\beta} B$ mit Sicherheitsfaktor β

Beispiel:

Bauchschm. re. u. \wedge Leukozyten > 10000 $\rightarrow_{0.6}$ Blinddarmentzündung

- Formeln für die Verknüpfung der Faktoren von Regeln
- Kalkül inkorrekt
- inkonsistente Ergebnisse konnten abgeleitet werden.

Andere Formalismen zur Modellierung von Unsicherheit

- nichtmonotone Logiken
- Defaultlogik
- Dempster-Schäfer-Theorie: ordnet einer logischen Aussage A eine Glaubensfunktion (engl. belief function) $Bel(A)$ zu
- Fuzzy-Logik (\Rightarrow Regelungstechnik)

Schließen mit bedingten Wahrscheinlichkeiten

- bedingte Wahrscheinlichkeiten statt Implikation (materiale Implikation)
- subjektive Wahrscheinlichkeiten
- Wahrscheinlichkeitstheorie sehr gut fundiert
- Schließen mit unsicherem und unvollständigem Wissen
- Methode der maximalen Entropie (MaxEnt)
- Bayes-Netze

Rechnen mit Wahrscheinlichkeiten

Beispiel:

- Wahrscheinlichkeit für „Würfeln einer Sechs“ gleich $1/6$
- Wahrscheinlichkeit für „Würfeln einer ungeraden Zahl“ gleich $1/2$

Definition 7.2 Sei Ω die zu einem Versuch gehörende endliche Menge von **Ereignissen**. Jedes Ereignis $\omega \in \Omega$ steht für einen möglichen Ausgang des Versuchs. Schließen sich die Ereignisse $w_i \in \Omega$ gegenseitig aus, decken aber alle möglichen Ausgänge des Versuchs ab, so werden diese **Elementarereignisse** genannt.

- Einmaliges Würfeln:

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

- Würfeln einer geraden Zahl $\{2, 4, 6\}$ ist kein Elementarereignis
- Würfeln einer Zahl kleiner als 5 $\{1, 2, 3, 4\}$ ist kein Elementarereignis
- Grund: $\{2, 4, 6\} \cap \{1, 2, 3, 4\} = \{2, 4\} \neq \emptyset$
- Mit zwei Ereignissen A und B ist $A \cup B$ ein Ereignis.
- Ω ist das **sichere Ereignis**
- die leere Menge \emptyset ist das **unmögliches Ereignis**
- statt $A \cap B$ schreiben wir $A \wedge B$, denn

$$x \in A \cap B \Leftrightarrow x \in A \wedge x \in B.$$

Mengenschreibweise	Aussagenlogik	Beschreibung
$A \cap B$	$A \wedge B$	Schnittmenge / und
$A \cup B$	$A \vee B$	Vereinigung / oder
\bar{A}	$\neg A$	Komplement / Negation
Ω	w	sicheres Ereignis / wahr
\emptyset	f	unmögliches Ereignis / falsch

- A, B , etc.: **Zufallsvariablen**
- hier nur diskrete Zufallsvariablen mit endlichem Wertebereich
- *Augenzahl* ist diskret mit den Werten 1,2,3,4,5,6.
- Wahrscheinlichkeit, 5 oder 6 zu würfeln ist gleich 1/3:

$$P(\text{Augenzahl} \in \{5, 6\}) = P(\text{Augenzahl} = 5 \vee \text{Augenzahl} = 6) = 1/3.$$

Definition 7.4 Sei $\Omega = \{\omega_1, \omega_2, \dots, \omega_n\}$ endlich. Es sei kein Elementereignis bevorzugt, d.h. man setzt eine Symmetrie bezüglich der Häufigkeit des Auftretens aller Elementarereignisse voraus. Die **Wahrscheinlichkeit** $P(A)$ des Ereignisses A ist dann

$$P(A) = \frac{|A|}{|\Omega|} = \frac{\text{Anzahl der für } A \text{ günstigen Fälle}}{\text{Anzahl der möglichen Fälle}}$$

Beispiel: Die Wahrscheinlichkeit, eine gerade Augenzahl zu würfeln ist

$$P(\text{Augenzahl} \in \{2, 4, 6\}) = \frac{|\{2, 4, 6\}|}{|\{1, 2, 3, 4, 5, 6\}|} = \frac{3}{6} = \frac{1}{2}.$$

- Jedes Elementarereignis hat die Wahrscheinlichkeit $1/|\Omega|$
(Laplace-Annahme)
- geht nur für endliche Ereignisträume
- Beispiel: *Augenfarbe* mit Werten *grün*, *blau*, *braun*
- *Augenfarbe = blau* beschreibt
- binären (booleschen) Variablen sind selbst eine Aussagen
- Beispiel: $P(JohnRuftAn)$ statt $P(JohnRuftAn = w)$

Direkt aus der Definition folgen einige Regeln:

Satz 7.2

1. $P(\Omega) = 1$.
2. $P(\emptyset) = 0$, d.h. das unmögliche Ereignis hat die Wahrscheinlichkeit 0.
3. Für paarweise unvereinbare Ereignisse A und B gilt $P(A \vee B) = P(A) + P(B)$.
4. Für zwei zueinander komplementäre Ereignisse A und $\neg A$ gilt $P(A) + P(\neg A) = 1$.
5. Für beliebige Ereignisse A und B gilt $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$.
6. Für $A \subseteq B$ gilt $P(A) \leq P(B)$.
7. Sind A_1, \dots, A_n die Elementarereignisse, so gilt $\sum_{i=1}^n P(A_i) = 1$ (Normierungsbedingung).

Beweis als Übung

Wahrscheinlichkeitsverteilung (Verteilung) für A, B

$$\mathbf{P}(A, B) = (P(A, B), P(A, \neg B), P(\neg A, B), P(\neg A, \neg B))$$

Verteilung in Matrixform:

$\mathbf{P}(A, B)$	$B = w$	$B = f$
$A = w$	$P(A, B)$	$P(A, \neg B)$
$A = f$	$P(\neg A, B)$	$P(\neg A, \neg B)$

$$P(A, B) = P(A \wedge B)$$

Wahrscheinlichkeitsverteilung

- (engl. joint probability distribution) :
- d Variablen X_1, \dots, X_d mit je n Werten:
- Verteilung enthält Werte $P(X_1 = x_1, \dots, X_d = x_d)$
- x_1, \dots, x_d nehmen jeweils n verschiedene Werte an.
- d -dimensionale Matrix mit insgesamt n^d Elementen darstellen.
- einer der n^d Werte ist redundant
- Verteilung wird durch $n^d - 1$ Werte eindeutig charakterisiert.

Bedingte Wahrscheinlichkeiten

Beispiel: In der Doggenriedstraße in Weingarten wird die Geschwindigkeit von 100 Fahrzeugen gemessen.

Ereignis	Häufigkeit	rel. Häufigkeit
Fahrzeug beobachtet	100	1
Fahrer ist Student (S)	30	0.3
Geschwindigkeit zu hoch (G)	10	0.1
Fahrer ist Student u. Geschw. zu hoch ($S \wedge G$)	5	0.05

Fahren Studenten häufiger zu schnell als der Durchschnitt?

Antwort: **Bedingte Wahrscheinlichkeit**

$$P(G|S) = \frac{|\text{Fahrer ist Student und Geschw. zu hoch}|}{|\text{Fahrer ist Student}|} = \frac{5}{30} = \frac{1}{6} \approx 0.17$$

A-priori-Wahrscheinlichkeit $P(G) = 0.1$

Definition 7.6 Für zwei Ereignisse A und B ist die Wahrscheinlichkeit $P(A|B)$ für A unter der Bedingung B (**bedingte Wahrscheinlichkeit**) definiert durch

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

$P(A|B)$ = Wahrscheinlichkeit von A , wenn man nur das Ereignis B betrachtet, d.h.

$$P(A|B) = \frac{|A \wedge B|}{|B|}.$$

Beweis:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{\frac{|A \wedge B|}{|\Omega|}}{\frac{|B|}{|\Omega|}} = \frac{|A \wedge B|}{|B|}.$$

Definition 7.8 Gilt für zwei Ereignisse A und B

$$P(A|B) = P(A),$$

so nennt man diese Ereignisse unabhängig.

Satz 7.4 Für unabhängige Ereignisse A und B folgt aus der Definition

$$P(A \wedge B) = P(A) \cdot P(B).$$

Beweis?

Beispiel: Wahrscheinlichkeit für 2 Sechsen $1/36$, wenn die Würfel unabhängig sind, denn

$$P(W_1 = 6 \wedge W_2 = 6) = P(W_1 = 6) \cdot P(W_2 = 6) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36},$$

Fällt Würfel 2 immer gleich wie Würfel 1, so gilt

$$P(W_1 = 6 \wedge W_2 = 6) = \frac{1}{6}.$$

Die Kettenregel

Produktregel: $P(A \wedge B) = P(A|B)P(B)$

Kettenregel:

$$\begin{aligned} & \mathbf{P}(X_1, \dots, X_n) \\ &= \mathbf{P}(X_n | X_1, \dots, X_{n-1}) \cdot \mathbf{P}(X_1, \dots, X_{n-1}) \\ &= \mathbf{P}(X_n | X_1, \dots, X_{n-1}) \cdot \mathbf{P}(X_{n-1} | X_1, \dots, X_{n-2}) \cdot \mathbf{P}(X_1, \dots, X_{n-2}) \\ &= \mathbf{P}(X_n | X_1, \dots, X_{n-1}) \cdot \mathbf{P}(X_{n-1} | X_1, \dots, X_{n-2}) \cdot \dots \cdot \mathbf{P}(X_2 | X_1) \cdot \mathbf{P}(X_1) \\ &= \prod_{i=1}^n \mathbf{P}(X_i | X_1, \dots, X_{i-1}), \end{aligned} \tag{7.1}$$

Marginalisierung

für zweiwertige Variablen A und B gilt

$$P(A) = P((A \wedge B) \vee (A \wedge \neg B)) = P(A \wedge B) + P(A \wedge \neg B).$$

allgemein:

$$P(X_1 = x_1, \dots, X_{d-1} = x_{d-1}) = \sum_{x_d} P(X_1 = x_1, \dots, X_{d-1} = x_{d-1}, X_d = x_d).$$

- Die resultierende Verteilung $\mathbf{P}(X_1, \dots, X_{d-1})$ heißt **Randverteilung**
- Projektion eines Quaders auf eine Seitenfläche.

Beispiel:

Leuko : Leukozytenwert größer als 10000

App : Patient hat Appendizitis (akut entzündeten Blinddarm),

$\mathbf{P}(App, Leuko)$	App	$\neg App$	gesamt
<i>Leuko</i>	0.23	0.31	0.54
$\neg Leuko$	0.05	0.41	0.46
gesamt	0.28	0.72	1

z.B. gilt:

$$P(Leuko) = P(App, Leuko) + P(\neg App, Leuko) = 0.54.$$

$$P(Leuko|App) = \frac{P(Leuko, App)}{P(App)} = 0.82$$

Die Bayes-Formel

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} \quad \text{sowie} \quad P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

Bayes-Formel:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{7.2}$$

Appendizitis-Beispiel:

- $P(App|Leuko)$ wäre für die Diagnose der Appendizitis interessanter, wird aber nicht publiziert!
- Warum wird $P(Leuko|App)$ publiziert, $P(App|Leuko)$ aber nicht?

$$P(App|Leuko) = \frac{P(Leuko|App) \cdot P(App)}{P(Leuko)} = \frac{0.82 \cdot 0.28}{0.54} = 0.43 \tag{7.3}$$

Die Methode der Maximalen Entropie

- Kalkül zum Schließen bei unsicherem Wissen
- Oft zu wenig Wissen für die Lösung der notwendigen Gleichungen
- Idee von E.T. Jaynes (Physiker):
Entropie der gesuchten Wahrscheinlichkeitsverteilung maximieren! [Jay57, Jay03]
[Che83, Nil86, Kan89, KK92] und
- Anwendung am Beispiel des LEXMED-Projektes

Eine Inferenzregel für Wahrscheinlichkeiten

Modus Ponens:

$$\frac{A, A \rightarrow B}{B}$$

Verallgemeinerung auf Wahrscheinlichkeitsregeln

$$\frac{P(A) = \alpha, \quad P(B|A) = \beta}{P(B) = ?}$$

Gegeben: zwei Wahrscheinlichkeitswerte α, β , **gesucht:** $P(B)$

Marginalisierung:

$$P(B) = P(A, B) + P(\neg A, B) = P(B|A) \cdot P(A) + P(B|\neg A) \cdot P(\neg A).$$

Mit klassischer Wahrscheinlichkeitsrechnung nur: $P(B) \geq P(B|A) \cdot P(A)$.

Verteilung

$$\mathbf{P}(A, B) = (P(A, B), P(A, \neg B), P(\neg A, B), P(\neg A, \neg B))$$

Abkürzung

$$\begin{aligned} p_1 &= P(A, B) \\ p_2 &= P(A, \neg B) \\ p_3 &= P(\neg A, B) \\ p_4 &= P(\neg A, \neg B) \end{aligned}$$

- Diese vier Parameter bestimmen die Verteilung.
- Daraus lässt sich jede Wahrscheinlichkeit für A und B berechnen.
- Vier Gleichungen werden benötigt.

Normierungsbedingung: $p_1 + p_2 + p_3 + p_4 = 1$

$$P(A, B) = P(B|A) \cdot P(A) = \alpha\beta$$

$$P(A) = P(A, B) + P(A, \neg B)$$

Gleichungssystem:

$$p_1 = \alpha\beta \tag{7.4}$$

$$p_1 + p_2 = \alpha \tag{7.5}$$

$$p_1 + p_2 + p_3 + p_4 = 1 \tag{7.6}$$

$$(7.4) \text{ in } (7.5): \quad p_2 = \alpha - \alpha\beta = \alpha(1 - \beta) \tag{7.7}$$

$$(7.5) \text{ in } (7.6): \quad p_3 + p_4 = 1 - \alpha \tag{7.8}$$

eine Gleichung fehlt!

Lösung eines Optimierungsproblems.

Gesucht: Verteilung \mathbf{p} für p_3, p_4 , welche die Entropie

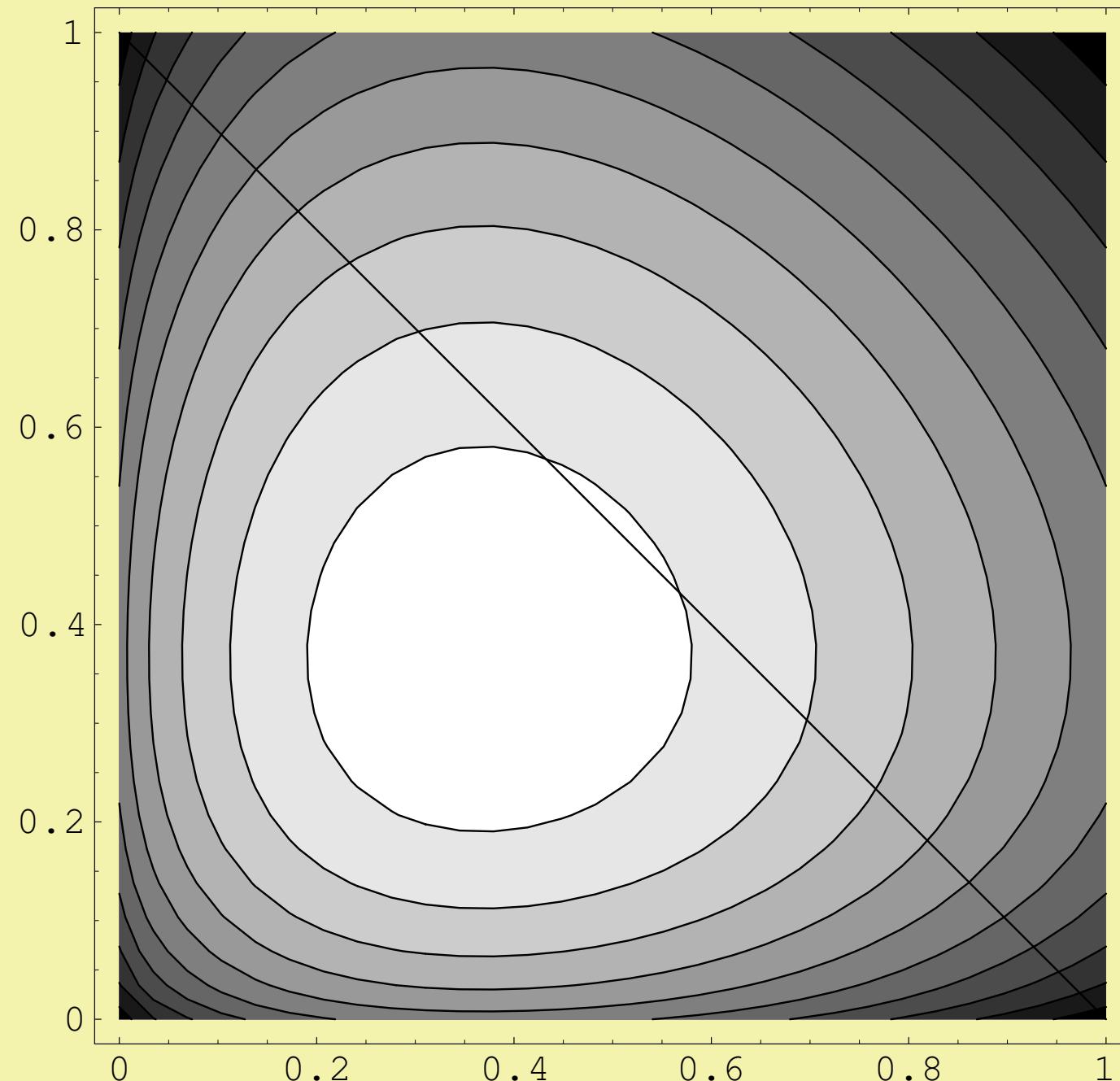
$$H(\mathbf{p}) = - \sum_{i=1}^n p_i \ln p_i = -p_3 \ln p_3 - p_4 \ln p_4$$

unter der Nebenbedingung $p_3 + p_4 = 1 - \alpha$ (Gleichung 7.8) maximiert.

Warum soll gerade die Entropiefunktion maximiert werden?

- Entropie misst die Unsicherheit einer Verteilung.
- negative Entropie als Maß für den Informationsgehalt der Verteilung.
- Maximieren der Entropie minimiert den Informationsgehalt der Verteilung.

Zweidimensionale Entropiefunktion mit Nebenbedingung $p_3 + p_4 = 1$.



Maximierung der Entropie

- Nebenbedingung: $p_3 + p_4 - 1 + \alpha = 0$
- Methode der Lagrangeparameter [BHW89]

Lagrangefunktion:

$$L = -p_3 \ln p_3 - p_4 \ln p_4 + \lambda(p_3 + p_4 - 1 + \alpha).$$

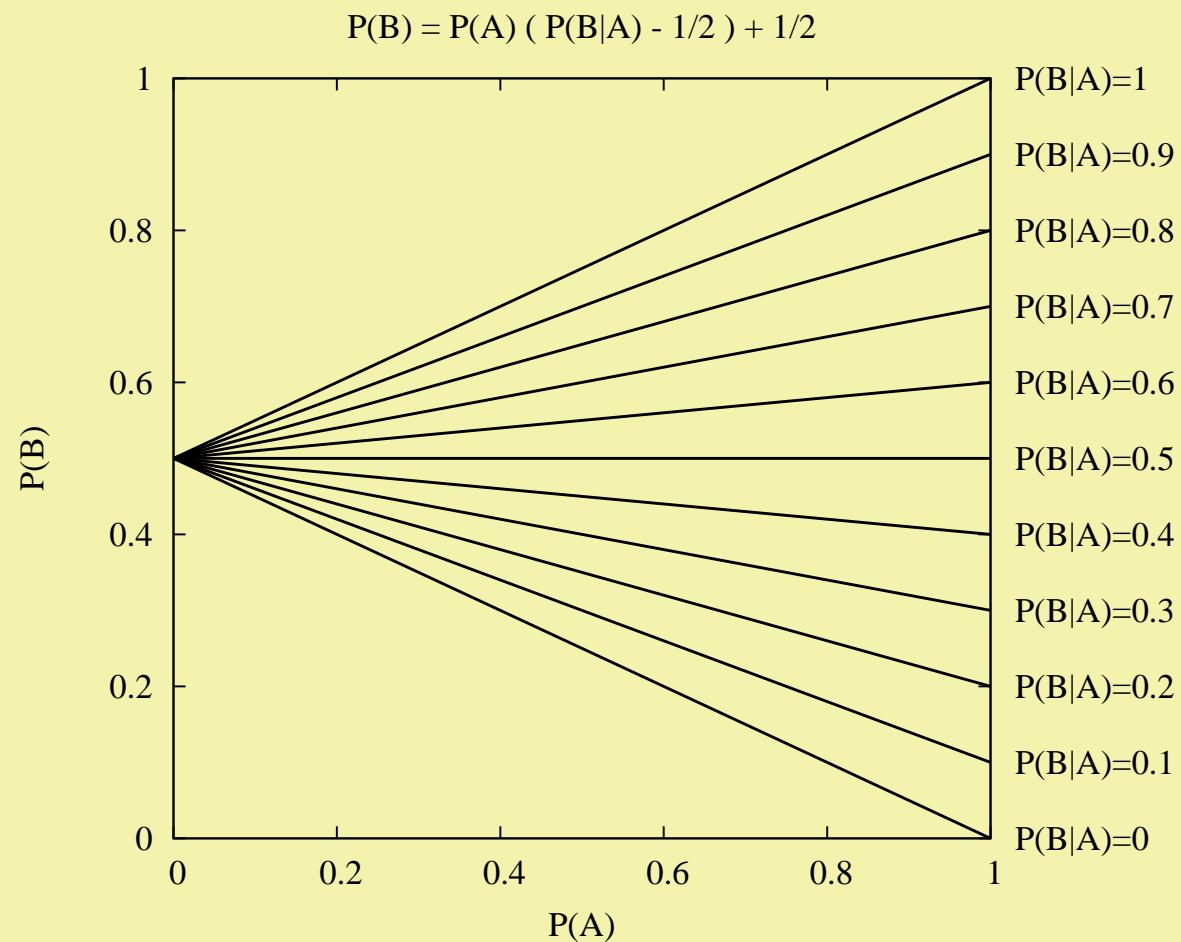
$$\frac{\partial L}{\partial p_3} = -\ln p_3 - 1 + \lambda = 0$$

$$\frac{\partial L}{\partial p_4} = -\ln p_4 - 1 + \lambda = 0$$

$$p_3 = p_4 = \frac{1 - \alpha}{2}.$$

$$P(B) = P(A, B) + P(\neg A, B) = p_1 + p_3 = \alpha\beta + \frac{1 - \alpha}{2} = \alpha(\beta - \frac{1}{2}) + \frac{1}{2}$$

Einsetzen von α und β : $P(B) = P(A)(P(B|A) - \frac{1}{2}) + \frac{1}{2}$.



Satz 7.6 Sei eine konsistente² Menge von linearen probabilistischen Gleichungen gegeben. Dann existiert ein eindeutiges Maximum der Entropiefunktion unter den gegebenen Gleichungen als Nebenbedingungen. Die dadurch definierte MaxEnt-Verteilung besitzt unter den Nebenbedingungen minimalen Informationsgehalt.

- Es gibt keine andere Verteilung, welche die Nebenbedingungen erfüllt und eine geringere Entropie hat.
- Ein Kalkül, der zu Verteilungen mit höherer Entropie führt, fügt ad hoc Informationen hinzu, was nicht gerechtfertigt ist.

²Eine Menge von probabilistischen Gleichungen heißt konsistent, wenn es mindestens eine Verteilung gibt, welche alle Gleichungen erfüllt.

- p_3 und p_4 kommen immer symmetrisch vor
- Daher $p_3 = p_4$ (Indifferenz)
- allgemein:

Definition 7.10 Wenn eine beliebige Vertauschung von zwei oder mehr Variablen in den Lagrangegleichungen diese in einen Satz äquivalenter Gleichungen überführt, so nennt man diese Variablen *indifferent*.

Satz 7.8 Ist eine Menge von Variablen $\{p_{i_1}, \dots, p_{i_k}\}$ *indifferent*, so liegt das Entropiemaximum unter den gegebenen Nebenbedingungen an einem Punkt mit $p_{i_1} = p_{i_2} = \dots = p_{i_k}$.

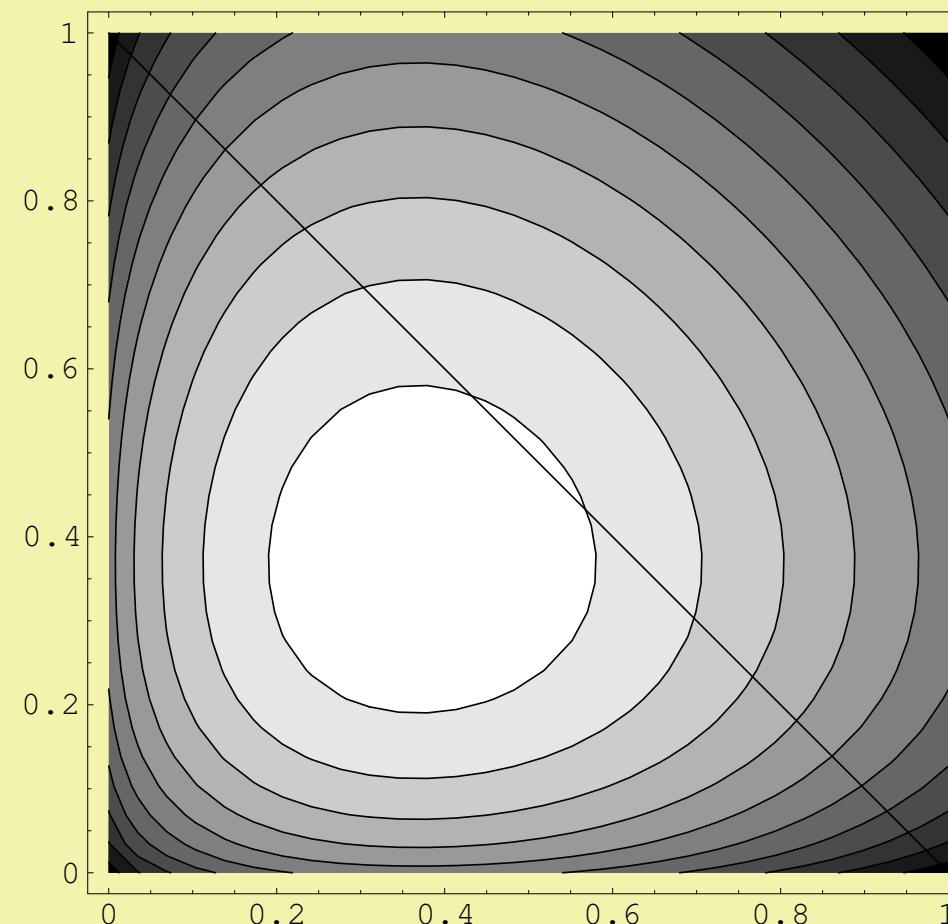
Entropiemaximum ohne explizite Nebenbedingungen

- gar kein Wissen gegeben
- außer der Normierungsbedingung $p_1 + p_2 + \dots + p_n = 1$ keine Nebenbedingungen.
- alle Variablen sind indifferent.
- $p_1 = p_2 = \dots = p_n = 1/n$. (Aufgabe ??)
- alle Welten gleich wahrscheinlich (Gleichverteilung)

Spezialfall: zwei Variablen A und B

$$P(A, B) = P(A, \neg B) = P(\neg A, B) = P(\neg A, \neg B) = 1/4,$$

$$P(A) = P(B) = 1/2 \quad \text{und} \quad P(B|A) = 1/2$$



Noch ein Beispiel

gegeben: $P(B|A) = \beta$

$$P(A, B) = P(B|A)P(A) = \beta P(A)$$

und

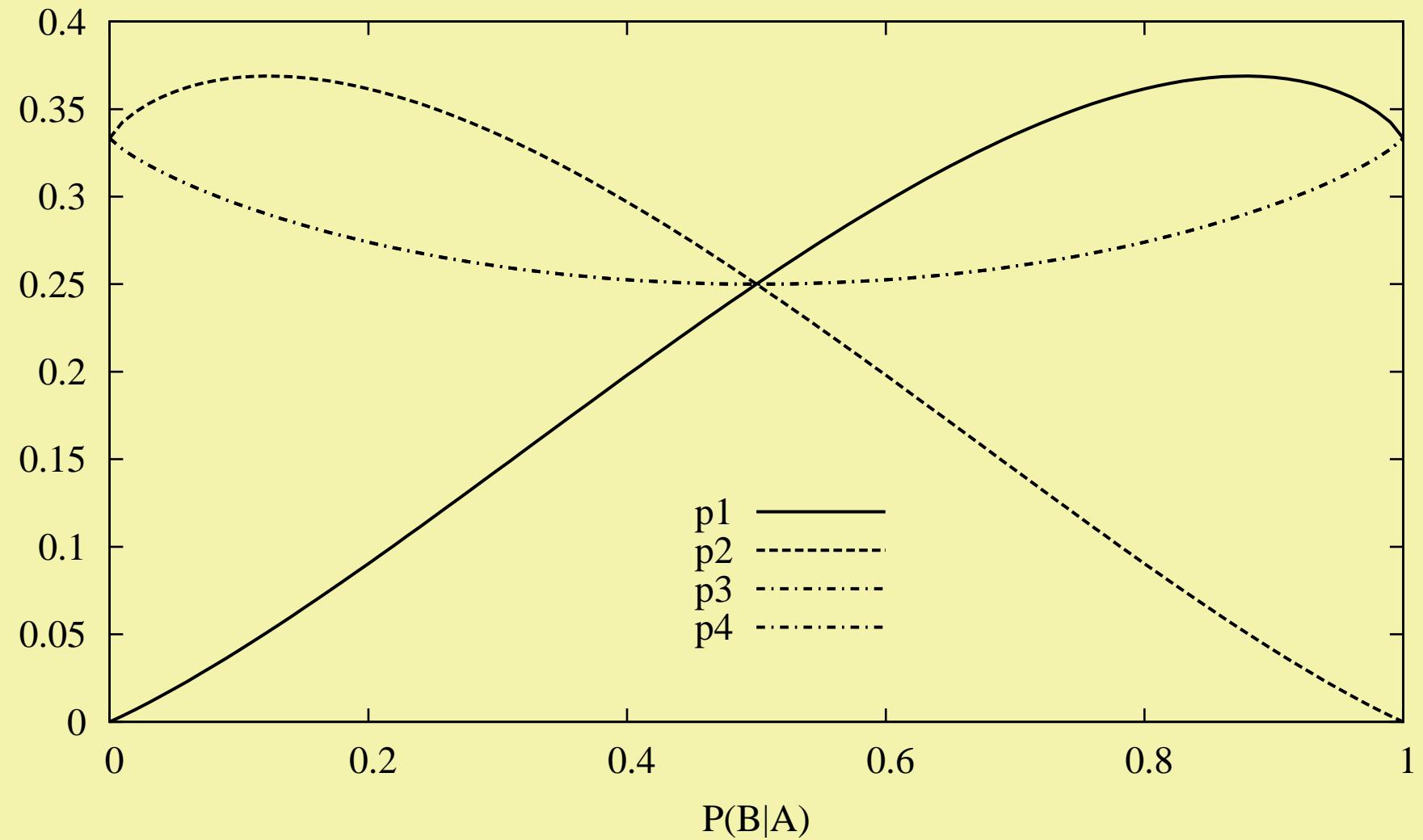
$$p_1 = \beta(p_1 + p_2)$$

Nebenbedingungen

$$\beta p_2 + (\beta - 1)p_1 = 0$$

$$p_1 + p_2 + p_3 + p_4 - 1 = 0.$$

- Keine symbolische Lösung!
- Numerisches Lösen der Lagrange-Gleichungen



p_1, p_2, p_3, p_4 in Abhängigkeit von β .

Bedingte Wahrscheinlichkeit versus materiale Implikation

A	B	$A \Rightarrow B$	$P(A)$	$P(B)$	$P(B A)$
w	w	w	1	1	1
w	f	f	1	0	0
f	w	w	0	1	<i>nicht definiert</i>
f	f	w	0	0	<i>nicht definiert</i>

Frage:

Welchen Wert nimmt $P(B|A)$ an, wenn nur $P(A) = \alpha$ und $P(B) = \gamma$ gegeben ist.

Bedingte Wahrscheinlichkeit versus materiale Implikation

$$p_1 = P(A, B), \quad p_2 = P(A, \neg B), \quad p_3 = P(\neg A, B), \quad p_4 = P(\neg A, \neg B)$$

Randbedingungen

$$p_1 + p_2 = \alpha \tag{7.9}$$

$$p_1 + p_3 = \gamma \tag{7.10}$$

$$p_1 + p_2 + p_3 + p_4 = 1 \tag{7.11}$$

Maximierung der Entropie (siehe Aufgabe ??):

$$p_1 = \alpha\gamma, \quad p_2 = \alpha(1 - \gamma), \quad p_3 = \gamma(1 - \alpha), \quad p_4 = (1 - \alpha)(1 - \gamma).$$

- Aus $p_1 = \alpha\gamma$ folgt $P(A, B) = P(A) \cdot P(B)$
- Unabhängigkeit von A und B .

Aus der Definition

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

folgt für $P(A) \neq 0$

$$P(B|A) = P(B)$$

Für $P(A) = 0$ bleibt $P(B|A)$ undefiniert.

A	B	$A \Rightarrow B$	$P(A)$	$P(B)$	$P(B A)$
w	w	w	α	β	β
f	w	w	0	β	<i>nicht definiert</i>

MaxEnt-Systeme

- MaxEnt-Optimierung meist nicht symbolisch durchführbar
- Daher: numerische Maximierung der Entropie
- SPIRIT: Fernuni Hagen [RM96,BK00]
- PIT: TU München [Sch96,ES99,SE00]
- **PIT** verwendet Sequential Quadratic Programming (SQP), um numerisch ein Extremum der Entropiefunktion zu finden.
- Als Eingabe erwartet PIT eine Datei mit den Randbedingungen:

```
var A{t,f}, B{t,f};  
  
P([A=t]) = 0.6;  
P([B=t] | [A=t]) = 0.3;  
  
  
QP([B=t]);  
QP([B=t] | [A=t]);
```

- Anfrage $QP([B=t])$

- Eingaben in www.pit-systems.de

Nr	Truthvalue	Probability	Query
1	UNSPECIFIED	3.800e-01	QP([B=t]);
2	UNSPECIFIED	3.000e-01	QP([A=t]- > [B=t]);

- $P(B) = 0.38$ und $P(B|A) = 0.3$ ab.

Das Tweety-Beispiel

$$P(\text{Vogel}|\text{Pinguin}) = 1$$

„Pinguine sind Vögel“

$$P(\text{Fliegt}|\text{Vogel}) \in [0.95, 1]$$

„(fast alle) Vögel können fliegen“

$$P(\text{Fliegt}|\text{Pinguin}) = 0$$

„Pinguine können nicht fliegen“

PIT-Eingabedatei:

```
var Pinguin{ja,nein}, Vogel{ja,nein}, Fliegt{ja,nein};

P([Vogel=ja] | [Pinguin=ja]) = 1;
P([Fliegt=ja] | [Vogel=ja]) IN [0.95,1];
P([Fliegt=ja] | [Pinguin=ja]) = 0;

QP([Fliegt=ja] | [Pinguin=ja]);
```

Antwort:

Nr	Truthvalue	Probability	Query
1	UNSPECIFIED	0.000e+00	QP([Pinguin=ja]- > [Fliegt=ja]);

MaxEnt und Nichtmonotonie

- Wahrscheinlichkeitsintervalle sind oft sehr hilfreich
- zweite Regel im Sinne von „normalerweise fliegen Vögel“:
 $P(Fliegt|Vogel) \in (0.5, 1]$
- MaxEnt ermöglicht nichtmonotones Schließen
- MaxEnt auch auf anspruchsvollen Benchmarks für nichtmonotones Schließen erfolgreich [Sch96]
- Anwendung von MaxEnt im medizinischen Expertensystem LEXMED

LEXMED, ein medizinisches Expertensystem für Appendizitisdiagnose

Manfred Schramm, Walter Rampf, Wolfgang Ertel

Hochschule Ravensburg-Weingarten,
Krankenhaus 14-Nothelfer, Weingarten
Technische Universität München

[SE00,Le99]

LEXMED = **lernfähiges Expertensystem für medizinische Diagnose.**

Das Projekt wurde finanziert vom Land Baden-Württemberg im Rahmen der Innovativen Projekte, von der AOK Baden-Württemberg, der Hochschule Ravensburg-Weingarten und vom Krankenhaus 14 Nothelfer in Weingarten.

Personenangaben	nicht bekannt	Werte	
Geschlecht	<input type="radio"/>	<input checked="" type="radio"/> männlich <input type="radio"/> weiblich	?
Altersgruppe	<input type="radio"/>	<input type="radio"/> 0-5 <input type="radio"/> 6-10 <input type="radio"/> 11-15 <input type="radio"/> 16-20 <input checked="" type="radio"/> 21-25 <input type="radio"/> 26-35 <input type="radio"/> 36-45 <input type="radio"/> <input type="radio"/> 46-55 <input type="radio"/> 56-65 <input type="radio"/> 65-	?
Untersuchungsergebnisse	nicht untersucht	Werte	
1. Schmerzquadrant	<input checked="" type="radio"/>	<input type="radio"/> ja <input checked="" type="radio"/> nein	?
2. Schmerzquadrant	<input checked="" type="radio"/>	<input type="radio"/> ja <input type="radio"/> nein	?
3. Schmerzquadrant	<input type="radio"/>	<input checked="" type="radio"/> ja <input type="radio"/> nein	?
4. Schmerzquadrant	<input checked="" type="radio"/>	<input type="radio"/> ja <input type="radio"/> nein	?
Abwehrspannung	<input type="radio"/>	<input checked="" type="radio"/> lokal <input type="radio"/> global <input type="radio"/> keine	?
Loslassschmerz	<input type="radio"/>	<input checked="" type="radio"/> ja <input type="radio"/> nein	?
Erschütterungsschmerz	<input type="radio"/>	<input checked="" type="radio"/> ja <input type="radio"/> nein	?
Rektalschmerz	<input checked="" type="radio"/>	<input type="radio"/> ja <input type="radio"/> nein	?
Darmgeräusche	<input type="radio"/>	<input type="radio"/> schwach <input checked="" type="radio"/> normal <input type="radio"/> vermehrt <input type="radio"/> keine	?
Sonographisch auffällig	<input checked="" type="radio"/>	<input type="radio"/> ja <input type="radio"/> nein	?
Pathologisches Urinsediment	<input checked="" type="radio"/>	<input type="radio"/> ja <input type="radio"/> nein	?
Rektaler Temperaturbereich	<input type="radio"/>	<input type="radio"/> -37.3 <input type="radio"/> 37.4-37.6 <input type="radio"/> 37.7-38.0 <input type="radio"/> 38.1-38.4 <input checked="" type="radio"/> 38.5-38.9 <input type="radio"/> 39.0-	?
Leukozytenbereich	<input type="radio"/>	<input type="radio"/> 0-6k <input type="radio"/> 6k-8k <input type="radio"/> 8k-10k <input type="radio"/> 10k-12k <input checked="" type="radio"/> 12k-15k <input type="radio"/> 15k-20k <input type="radio"/> 20k-	?
Abfragen			
Diagnose(4w)	?	Diagnose(3w)	?
		Datenbankabfrage	?

Ergebnis der PIT-Diagnose

Diagnose	App. entzündet	App. perforiert	negativ	vandere
Wahrscheinlichkeit	0.70	0.17	0.06	0.07

Appendizitisdiagnose mit formalen Methoden

- häufigste ernsthafte Ursache für akute Bauchschmerzen [Dom91]: Appendizitis
- schwierige Diagnose [Ohm95].
- ca. 20% der entfernten Appendizes unauffällig
- Ebenso gibt es Fälle, in denen ein entzündeter Appendix nicht erkannt wird.
- de Dombal (Großbritannien) entwickelte 1972 ein Expertensystem zur Diagnose akuter Bauchschmerzen. [Dom72, Ohm94, Ohm95].

Lineare Scores

Für jeden Wert eines Symptoms (zum Beispiel **Fieber** oder **Bauchschmerzen rechts unten**) notiert der Arzt eine bestimmte Anzahl an Punkten. Bei n Symptomen S_1, \dots, S_n lässt sich ein Score formal als

$$\text{Diagnose} = \begin{cases} \text{Appendizitis falls } w_1S_1 + \dots + w_nS_n > \Theta \\ \text{negativ sonst} \end{cases}$$

beschreiben.

- Scores sind zu schwach für die Modellierung komplexer Zusammenhänge.
- Scoresysteme können keine „Kontexte“ berücksichtigen.
- Sie können z.B. nicht zwischen dem Leukozytenwert eines älteren Patienten und dem eines Jüngeren unterscheiden.
- stellen hohe Anforderungen an Datenbanken (repräsentativ).

Hybride Probabilistische Wissensbasis

Anfrage an das Expertensystem:

Wie hoch ist die Wahrscheinlichkeit für einen entzündeten Appendix, wenn der Patient ein 23-jähriger Mann mit Schmerzen im rechten Unterbauch und einem Leukozytenwert von 13000 ist?

$$P(Bef4 = \text{entzündet} \vee Bef4 = \text{perforiert} \mid \\ Geschl = \text{männlich} \wedge Alter \in 21\text{-}25 \wedge Leuko \in 12k\text{-}15k).$$

Symptom	Werte	#	Abk.
Geschlecht	männlich, weiblich	2	Geschl
Alter	0-5, 6-10, 11-15, 16-20, 21-25, 26-35, 36-45, 46-55, 56-65, 65-	10	Alter
Schmerz 1. Quad.	ja, nein	2	S1Q
Schmerz 2. Quad.	ja, nein	2	S2Q
Schmerz 3. Quad.	ja, nein	2	S3Q
Schmerz 4. Quad.	ja, nein	2	S4Q
Abwehrspannung	lokal, global, keine	3	Abw
Loslassschmerz	ja, nein	2	Losl
S. bei Erschütterung	ja, nein	2	Ersch
Rektalschmerz	ja, nein	2	RektS
Darmgeräusche	schwach, normal, vermehrt, keine	4	Darmg
Pos. Sonographiebef.	ja, nein	2	Sono
Path. Urinsedim.	ja, nein	2	PathU
Fieber rektal	-37.3, 37.4-37.6, 37.7-38.0, 38.1-38.4, 38.5-6 38.9, 39.0-	6	TRek
Leukozyten	0-6k, 6k-8k, 8k-10k, 10k-12k, 12k-15k, 15k-7 20k, 20k-	7	Leuko
Befund	entzündet, perforiert, negativ, andere	4	Bef4

Wissensquellen

Datenbank

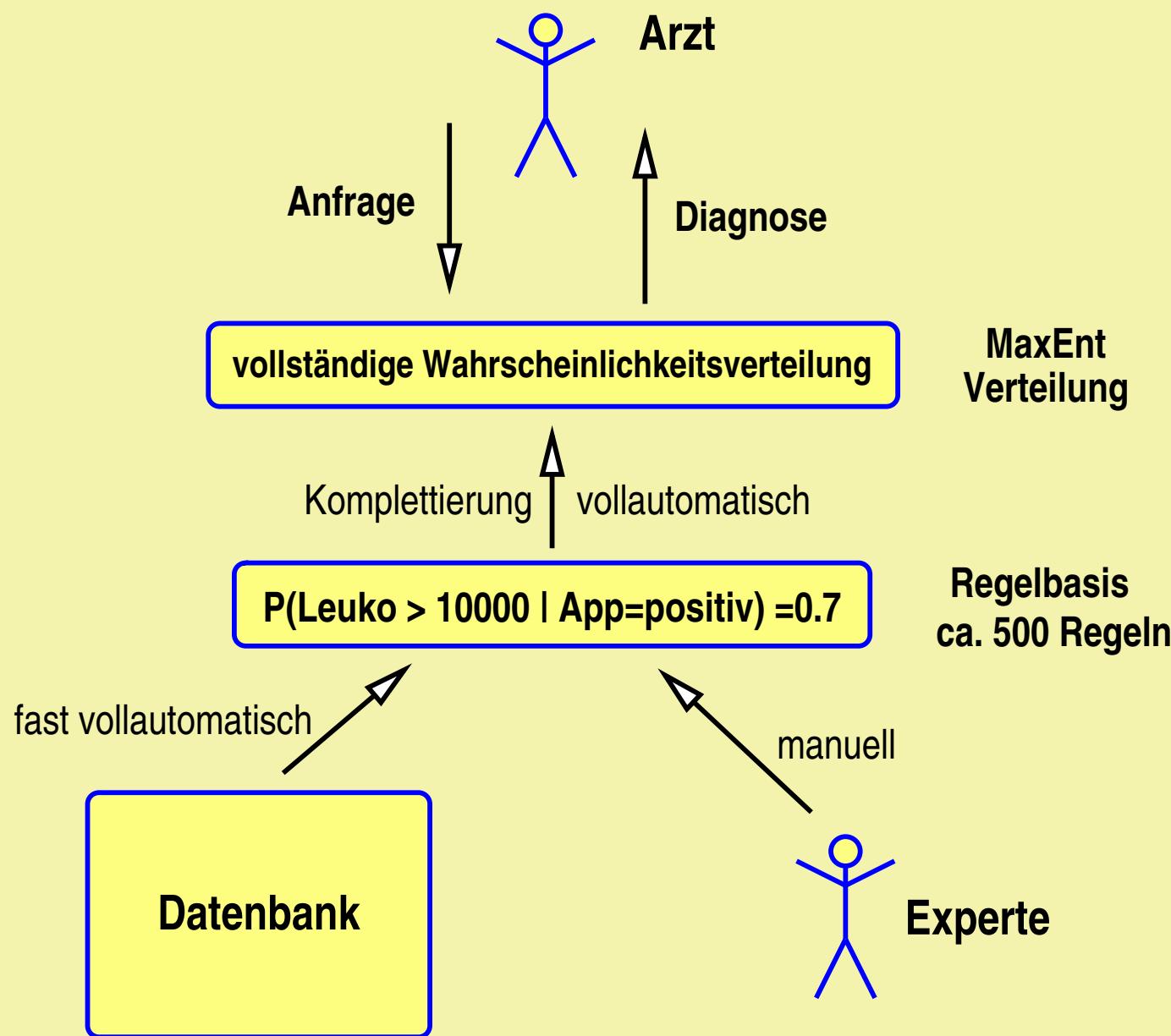
15000 Patienten aus
Baden-Württemberg 1995

Expertenwissen

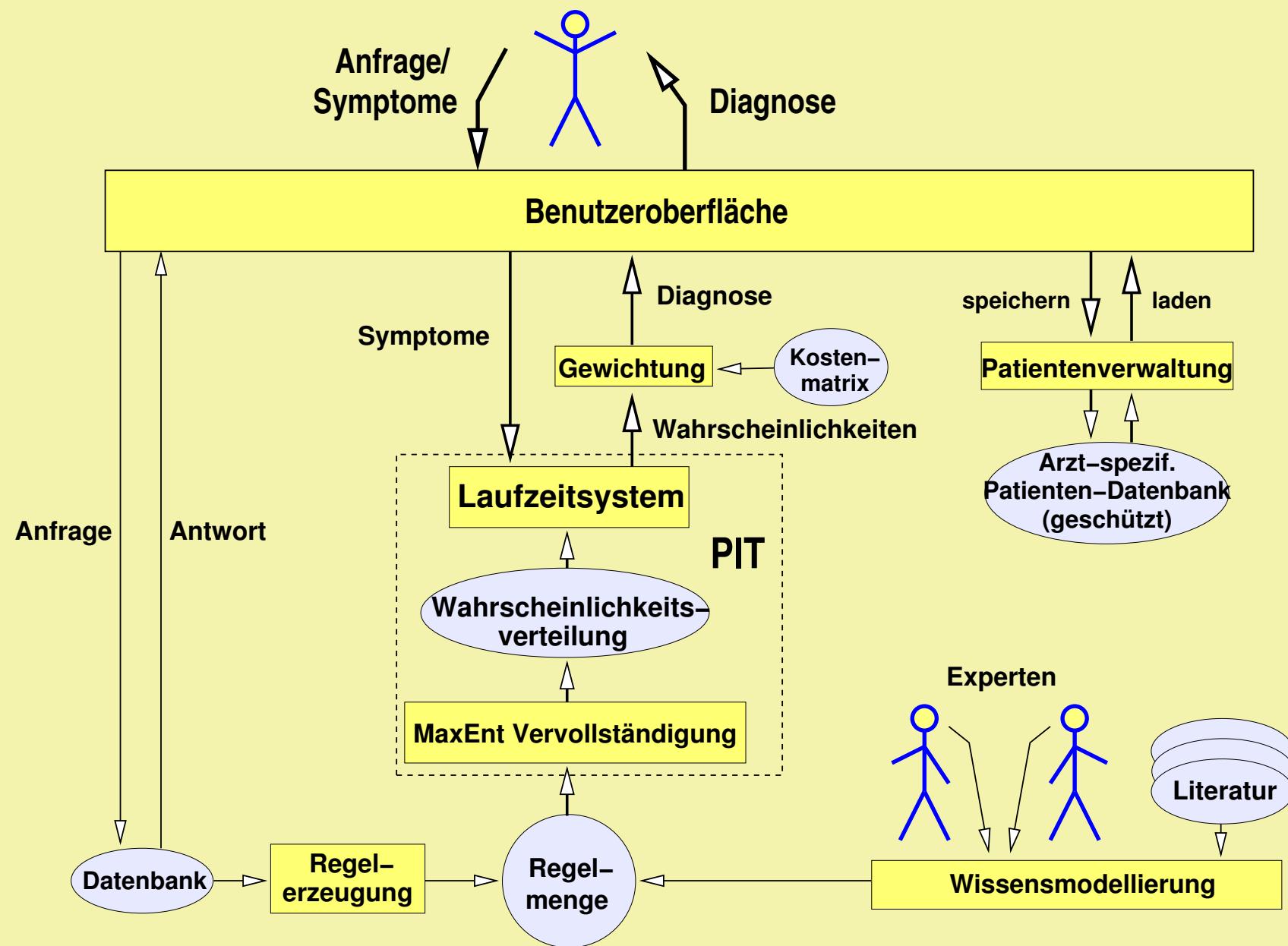
Dr. Rampf

Dr. Hontschik

Wissensverarbeitung



Systemarchitektur



Wahrscheinlichkeitsverteilung

- Mächtigkeit der Verteilung:

$$2^{10} \cdot 10 \cdot 3 \cdot 4 \cdot 6 \cdot 7 \cdot 4 = 20\,643\,840$$

- 20 643 839 unabhängige Werte.
- Jede Regelmenge mit weniger als 20 643 839 Wahrscheinlichkeitswerten beschreibt diesen Ereignisraum eventuell nicht vollständig.
- vollständige Verteilung wird benötigt.
- menschlicher Experte kann nicht 20 643 839 Werte liefern!

Funktion von LEXMED

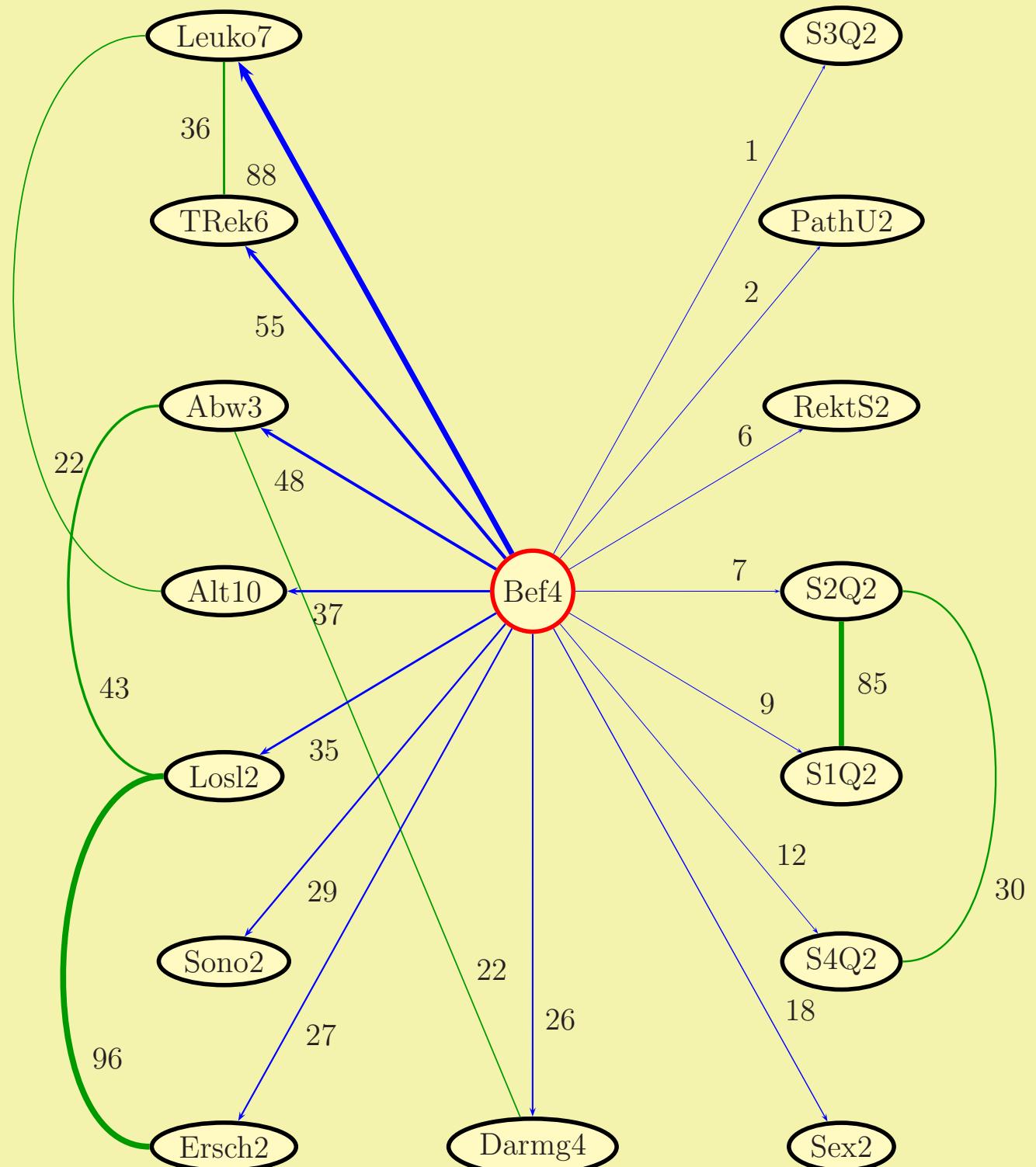
Wahrscheinlichkeitsaussagen:

$$P(\text{Leuko} > 20000 \mid \text{Bef4} = \text{entzündet}) = 0.09$$

3

³Statt einzelner numerischer Werte können hier auch Intervalle (beispielsweise [0.06, 0.12]) verwendet werden.

Der Abhängigkeitsgraph



Lernen von Regeln durch Statistische Induktion

- Schätzen der Regelwahrscheinlichkeiten
- Struktur des Abhängigkeitsgraphen = Struktur der gelernten Regeln (wie Bayes-Netz)
- A-priori-Regeln, z.B. Gleichung 7.12),
- Regeln mit einfacher Bedingung, z.B. Gleichung 7.13
- Regeln: Befund und 2 Symptome (Gleichung 7.14)

$$P(Bef4 = \text{entzündet}) = 0.40 \quad (7.12)$$

$$P(Sono = ja | Bef4 = \text{entzündet}) = 0.43 \quad (7.13)$$

$$P(S4Q = ja | Bef4 = \text{entzündet} \wedge S2Q = ja) = 0.61 \quad (7.14)$$

```

1 P([Leuko7=0-6k] | [Bef4=negativ] * [Alt10=16-20]) = [0.132,0.156];
2 P([Leuko7=6-8k] | [Bef4=negativ] * [Alt10=16-20]) = [0.257,0.281];
3 P([Leuko7=8-10k] | [Bef4=negativ] * [Alt10=16-20]) = [0.250,0.274];
4 P([Leuko7=10-12k] | [Bef4=negativ] * [Alt10=16-20]) = [0.159,0.183];
5 P([Leuko7=12-15k] | [Bef4=negativ] * [Alt10=16-20]) = [0.087,0.112];
6 P([Leuko7=15-20k] | [Bef4=negativ] * [Alt10=16-20]) = [0.032,0.056];
7 P([Leuko7=20k-] | [Bef4=negativ] * [Alt10=16-20]) = [0.000,0.023];
8 P([Leuko7=0-6k] | [Bef4=negativ] * [Alt10=21-25]) = [0.132,0.172];
9 P([Leuko7=6-8k] | [Bef4=negativ] * [Alt10=21-25]) = [0.227,0.266];
10 P([Leuko7=8-10k] | [Bef4=negativ] * [Alt10=21-25]) = [0.211,0.250];
11 P([Leuko7=10-12k] | [Bef4=negativ] * [Alt10=21-25]) = [0.166,0.205];
12 P([Leuko7=12-15k] | [Bef4=negativ] * [Alt10=21-25]) = [0.081,0.120];
13 P([Leuko7=15-20k] | [Bef4=negativ] * [Alt10=21-25]) = [0.041,0.081];
14 P([Leuko7=20k-] | [Bef4=negativ] * [Alt10=21-25]) = [0.004,0.043];

```

Einige LEXMED-Regeln mit Wahrsch.-Intervallen. „*“ steht für „ \wedge “.

Wahrscheinl. schätzen durch Zählen der Häufigkeit:

$$\frac{|Bef4 = \text{entzündet} \wedge Sono = ja|}{|Bef4 = \text{entzündet}|} = 0.43.$$

Risikomanagement mit Hilfe der Kostenmatrix

		Ergebnis der PIT-Diagnose			
Diagnose	App. entzündet		App. perforiert	negativ	andere
Wahrscheinlichkeit	0.24	0.16	0.55	0.05	

Was tun?

Die Kostenmatrix

		Wahrscheinlichkeit für versch. Befunde			
		entzündet	perforiert	negativ	andere
Therapie	Therapie	0.25	0.15	0.55	0.05
	Operation	0	500	5800	6000
Not-Operation		500	0	6300	6500
Ambulant beob.		12000	150000	0	16500
Sonstiges		3000	5000	1300	0
Stationär beob.		3500	7000	400	600
					2175

- Optimale Entscheidungen haben (Mehr-)Kosten 0.
- Ziel: Therapie mit den minimalen mittleren Kosten.
- Wahrscheinlichkeitsvektor = $(0.25, 0.15, 0.55, 0.05)$
- zu erwartende Kosten für Fehlentscheidungen.
- Für erste Zeile: (*Operation*):
 $\text{Matrix} \cdot \text{Vektor} = 0.25 \cdot 0 + 0.15 \cdot 500 + 0.55 \cdot 5800 + 0.05 \cdot 6000 = 3565.$
- kostenorientierter Agent

Kostenmatrix im binären Fall

- Befunde: *Appendizitis* und *NSAP*
- Therapien: $P(\text{Operation}) = p_1$ und $P(\text{Ambulant beobachten}) = p_2$
- Kostenmatrix:

$$\begin{pmatrix} 0 & k_2 \\ k_1 & 0 \end{pmatrix}.$$

- **falsch positiv, Falsch negativ**

$$\begin{pmatrix} 0 & k_2 \\ k_1 & 0 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = (k_2 p_2, k_1 p_1)$$

$$1/k_1(k_2 p_2, k_1 p_1) = ((k_2/k_1)p_2, p_1)$$

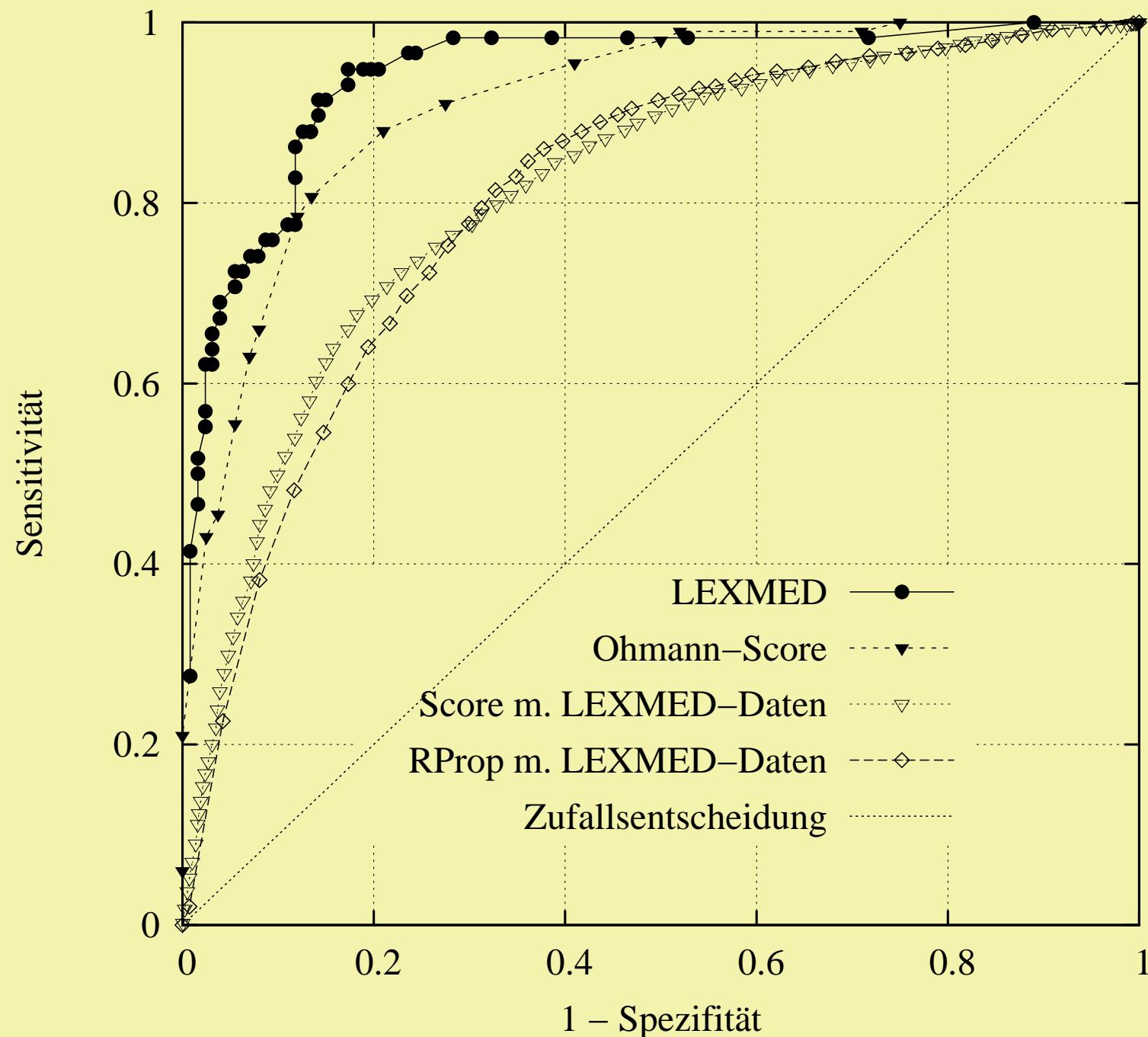
gleiches Ergebnis mit

$$\begin{pmatrix} 0 & k \\ 1 & 0 \end{pmatrix},$$

Risikomanagement.

- Arbeitspunkt des Diagnosesystems
- $k = 0$: System extrem riskant eingestellt
- $k \rightarrow \infty$ alle Patienten werden operiert.

Leistungsfähigkeit / ROC-Kurve



Einsatzgebiete und Erfahrungen

- Einsatz in der Diagnostik unterstützt
- Qualitätssicherung: Diagnosequalität von Krankenhäusern mit der von Expertensystemen vergleichen.
- seit 1999 im Einsatz im 14-Nothelfer-Krankenhaus Weingarten
- www.lexmed.de, ohne
- Diagnosequalität ist vergleichbar mit erfahrenem Chirurg
- sehr schwierig kommerziell zu vermarkten
- falscher Zeitpunkt?
- Wunsch d. Patienten nach persönlicher Betreuung!
- Seit de Dombal 1972 sind 35 Jahre vergangen. Wird es noch weitere 35 Jahre dauern?

Schließen mit Bayes-Netzen

- d Variablen X_1, \dots, X_d mit je n Werten
- Wahrscheinlichkeitsverteilung hat $n^d - 1$ Werte.
- in d. Praxis: Verteilung enthält viel Redundanz.

Unabhängige Variablen

$$\mathbf{P}(X_1, \dots, X_d) = \mathbf{P}(X_1) \cdot \mathbf{P}(X_2) \cdot \dots \cdot \mathbf{P}(X_d).$$

bedingte Wahrscheinlichkeiten werden trivial:⁴

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

An einem einfachen und sehr anschaulichen Beispiel von , das durch [Russell/Norvig](#) noch bekannter wurde und mittlerweile zum KI-Grundwissen gehört, wollen wir das Schließen mit Bayes-Netzen erläutern.

⁴Wird bei der Naive-Bayes-Methode mit Erfolg auf die Textklassifikation angewendet (siehe Abschnitt 8.5).

Das Alarm-Beispiel

J. Pearl Pearl Beispiel:

Wissensbasis:

$$\begin{array}{ll} P(J|AI) = 0.90 & P(M|AI) = 0.70 \\ P(J|\neg AI) = 0.05 & P(M|\neg AI) = 0.01 \end{array}$$

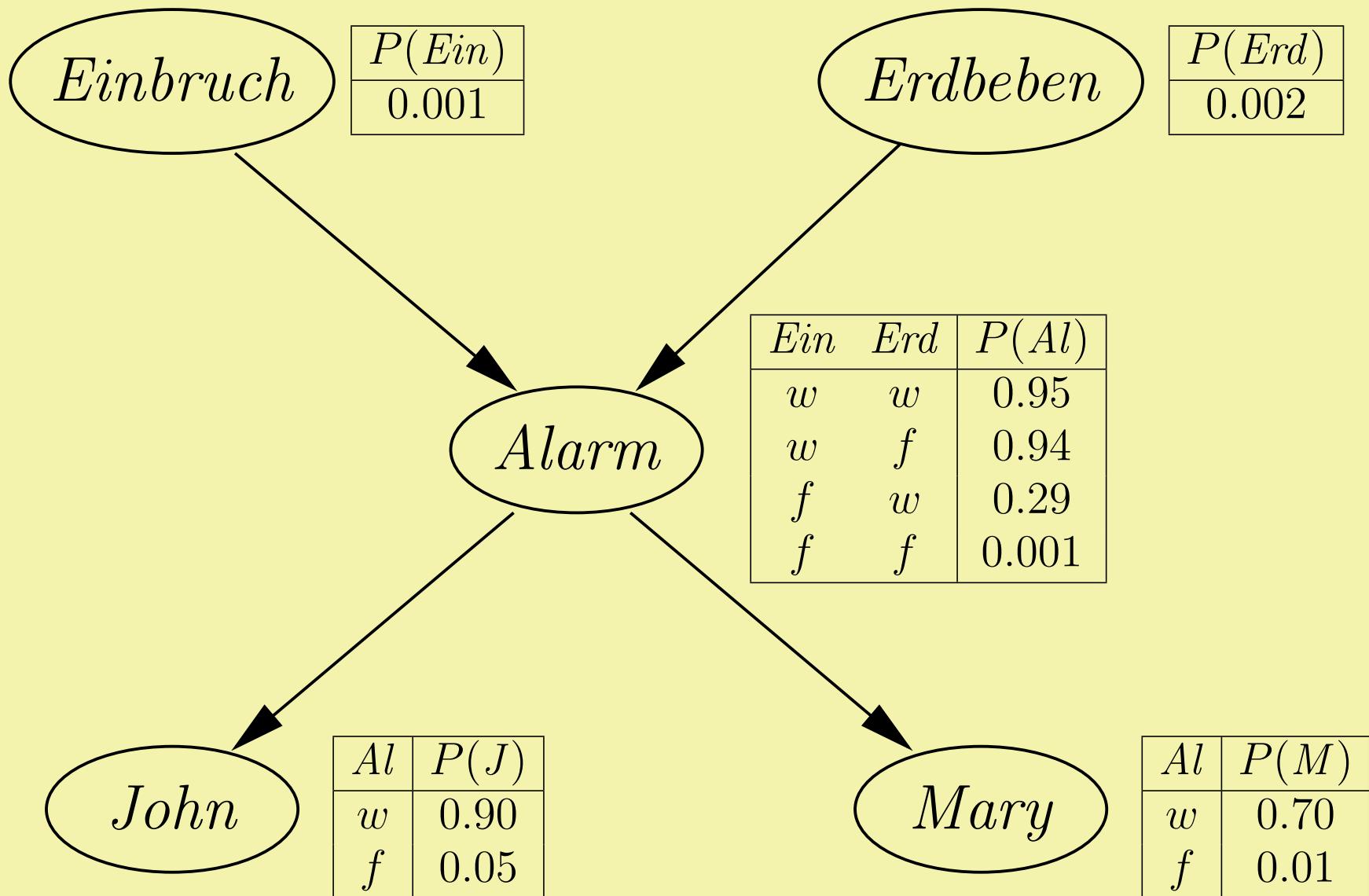
$$\begin{array}{l} P(AI|Ein, Erd) = 0.95 \\ P(AI|Ein, \neg Erd) = 0.94 \\ P(AI|\neg Ein, Erd) = 0.29 \\ P(AI|\neg Ein, \neg Erd) = 0.001, \end{array}$$

A-priori-Wahrscheinlichkeiten: $P(Ein) = 0.001$ $P(Erd) = 0.002$.

Anfragen: $P(Ein|J \vee M)$ $P(J|Ein)$ $P(M|Ein)$

J „John ruft an“ M „Mary ruft an“ AI , „Alarmsirene ertönt“
 Ein , „Einbruch“ Erd , „Erdbeben“

Graphische Darstellung des Wissens als Bayes-Netz



Bedingte Unabhängigkeit

Definition 7.12 Zwei Variablen A und B heißen *bedingt unabhängig*, gegeben C , wenn

$$P(A, B|C) = P(A|C) \cdot P(B|C).$$

Beispiele:

$$P(J, M|AI) = P(J|AI) \cdot P(M|AI)$$

$$P(J, Ein|AI) = P(J|AI) \cdot P(Ein|AI)$$

Bedingte Unabhängigkeit

Satz 7.10 Folgende Gleichungen sind paarweise äquivalent, das heißt jede einzelne dieser Gleichungen beschreibt die bedingte Unabhängigkeit der Variablen A und B gegeben C .

$$\mathbf{P}(A, B|C) = \mathbf{P}(A|C) \cdot \mathbf{P}(B|C) \quad (7.18)$$

$$\mathbf{P}(A|B, C) = \mathbf{P}(A|C) \quad (7.19)$$

$$\mathbf{P}(B|A, C) = \mathbf{P}(B|C) \quad (7.20)$$

Beweis:

$$\mathbf{P}(A, B, C) = \mathbf{P}(A, B|C)\mathbf{P}(C) = \mathbf{P}(A|C)\mathbf{P}(B|C)\mathbf{P}(C)$$

$$\mathbf{P}(A, B, C) = \mathbf{P}(A|B, C)\mathbf{P}(B|C)\mathbf{P}(C).$$

Also:

$$\mathbf{P}(A|B, C) = \mathbf{P}(A|C)$$

Praktische Anwendung

$$P(J|Ein) = \frac{P(J, Ein)}{P(Ein)} = \frac{P(J, Ein, AI) + P(J, Ein, \neg AI)}{P(Ein)} \quad (7.21)$$

$$\mathbf{P}(J, Ein, AI) = \mathbf{P}(J|Ein, AI)\mathbf{P}(AI|Ein)\mathbf{P}(Ein) = \mathbf{P}(J|AI)\mathbf{P}(AI|Ein)\mathbf{P}(Ein), \quad (7.22)$$

$$\begin{aligned} P(J|Ein) &= \frac{P(J|AI)P(AI|Ein)P(Ein) + P(J|\neg AI)P(\neg AI|Ein)P(Ein)}{P(Ein)} \\ &= P(J|AI)P(AI|Ein) + P(J|\neg AI)P(\neg AI|Ein). \end{aligned} \quad (7.23)$$

$P(AI|Ein)$ und $P(\neg AI|Ein)$ fehlen!

$$\begin{aligned}
P(AI|Ein) &= \frac{P(AI, Ein)}{P(Ein)} = \frac{P(AI, Ein, Erd) + P(AI, Ein, \negErd)}{P(Ein)} \\
&= \frac{P(AI|Ein, Erd)P(Ein)P(Erd) + P(AI|Ein, \negErd)P(Ein)P(\negErd)}{P(Ein)} \\
&= P(AI|Ein, Erd)P(Erd) + P(AI|Ein, \negErd)P(\negErd) \\
&= 0.95 \cdot 0.002 + 0.94 \cdot 0.998 = 0.94
\end{aligned}$$

$$P(\neg AI|Ein) = 0.06$$

$$P(J|Ein) = 0.9 \cdot 0.94 + 0.05 \cdot 0.06 = 0.849$$

$$\text{Analog: } P(M|Ein) = 0.659$$

$$P(J, M|Ein) = P(J|Ein)P(M|Ein) = 0.849 \cdot 0.659 = 0.559.$$

$$\begin{aligned}
 P(J \vee M | Ein) &= P(\neg(\neg J, \neg M) | Ein) = 1 - P(\neg J, \neg M | Ein) \\
 &= 1 - P(\neg J | Ein)P(\neg M | Ein) = 1 - 0.051 = 0.948.
 \end{aligned}$$

$$P(Ein | J) = \frac{P(J | Ein)P(Ein)}{P(J)} = \frac{0.849 \cdot 0.001}{0.052} = 0.016$$

$$P(Ein | M) = 0.056$$

$$P(Ein | J, M) = 0.284$$

(siehe Aufgabe ??).

$$P(J | Ein) = P(J | AI)P(AI | Ein) + P(J | \neg AI)P(\neg AI | Ein)$$

Konditionierung:

$$P(A|B) = \sum_c P(A|B, C=c)P(C=c|B).$$

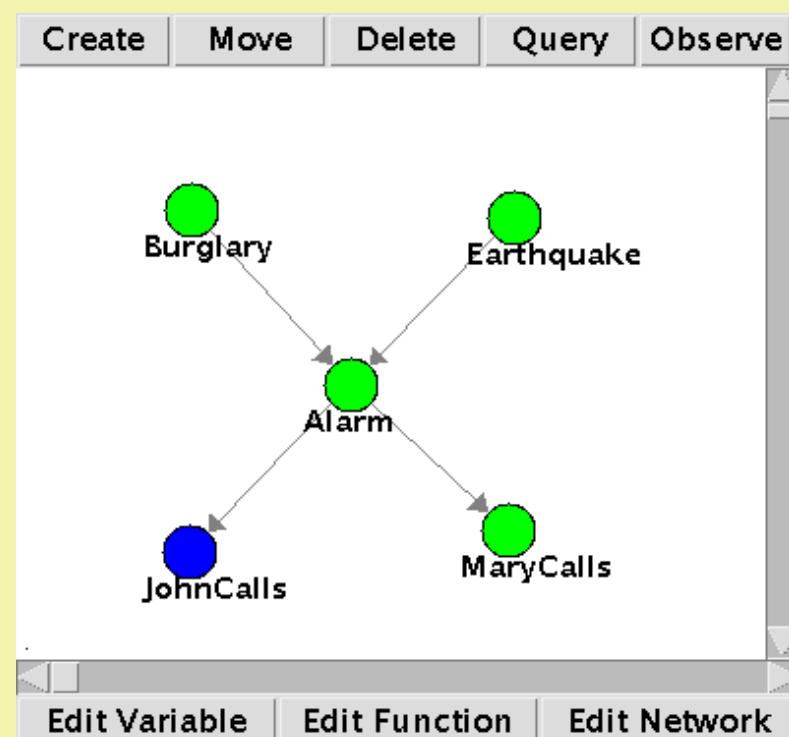
Software für Bayes-Netze: PIT

```
1 var Alarm{t,f}, Einbruch{t,f}, Erdbeben{t,f}, John{t,f}, Mary{t,f};  
2  
3 P([Erdbeben=t]) = 0.002;  
4 P([Einbruch=t]) = 0.001;  
5 P([Alarm=t] | [Einbruch=t] AND [Erdbeben=t]) = 0.95;  
6 P([Alarm=t] | [Einbruch=t] AND [Erdbeben=f]) = 0.94;  
7 P([Alarm=t] | [Einbruch=f] AND [Erdbeben=t]) = 0.29;  
8 P([Alarm=t] | [Einbruch=f] AND [Erdbeben=f]) = 0.001;  
9 P([John=t] | [Alarm=t]) = 0.90;  
10 P([John=t] | [Alarm=f]) = 0.05;  
11 P([Mary=t] | [Alarm=t]) = 0.70;  
12 P([Mary=t] | [Alarm=f]) = 0.01;  
13  
14 QP([Einbruch=t] | [John=t] AND [Mary=t]);
```

Antwort: $P([Einbruch=t] | [John=t] \text{ AND } [Mary=t]) = 0.2841$.

Man kann zeigen Schramm, dass bei der Angabe der CPTs oder äquivalenter Regeln aus dem MaxEnt-Prinzip die gleichen bedingten Unabhängigkeiten wie bei einem Bayes-Netz folgen.

Software für Bayes-Netze: JavaBayes



File Options Help

To query on a particular node, click on it.

To observe a node, click on it.

To query on a particular node, click on it.

Posterior distribution:
probability ("Burglary") { //1 variable(s) and 2 values
table
 0.9835562456661 // p(False | evidence)
 0.016443754333900076; // p(True | evidence);
}

Software für Bayes-Netze: Hugin

- mächtiger und komfortabler ist das
- professionell vertriebene System
- auch stetige Variablen möglich
- kann auch Bayes-Netze lernen, das heißt das Netz vollautomatisch aus statistischen Daten generieren

Entwicklung von Bayes-Netzen

Bei den Variablen v_1, \dots, v_n mit jeweils $|v_1|, \dots, |v_n|$ verschiedenen Werten hat die Verteilung insgesamt

$$\prod_{i=1}^n |v_i| - 1$$

unabhängige Einträge.

Alarm-Beispiel: $2^5 - 1 = 31$ unabhängige Einträge.

Bayes-Netz: Für einen Knoten v_i mit den k_i Elternknoten e_{i1}, \dots, e_{ik_i} besitzt die zugehörige CPT

$$(|v_i| - 1) \prod_{j=1}^{k_i} |e_{ij}|$$

Einträge.

Alle CPTs zusammen:

$$\sum_{i=1}^n (|v_i| - 1) \prod_{j=1}^{k_i} |e_{ij}| \quad (7.24)$$

⁵ Alarm-Beispiel:

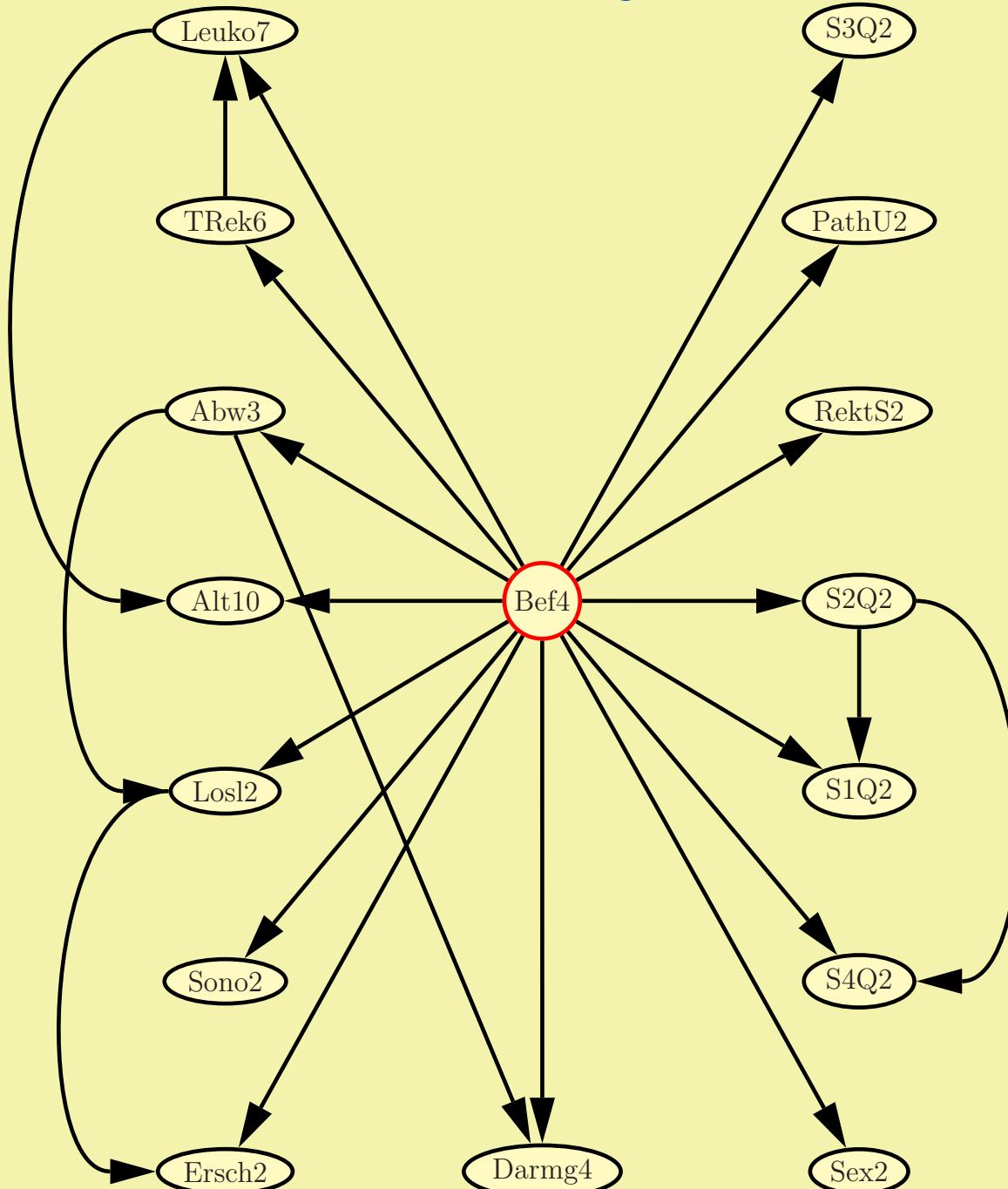
$$2 + 2 + 4 + 1 + 1 = 10$$

⁵Knoten ohne Vorgänger: Hierfür setzen wir den Wert 1 ein, denn die CPT enthält mit der A-priori-Wahrscheinlichkeit genau einen Wert.

Spezialfall

- n Variablen,
- gleiche Zahl b an Werten
- jeder Knoten hat k Elternknoten.
- Alle CPTs zusammen besitzen $n(b - 1)b^k$ Einträge.
- vollständige Verteilung enthält $b^n - 1$ Einträge.
- lokale Vernetzung
- Modularisierung

LEXMED als Bayes-Netz



LEXMED als Bayes-Netz

- Mächtigkeit der Verteilung: 20 643 839
- Mächtigkeit des Bayes-Netzes: 521 Werte

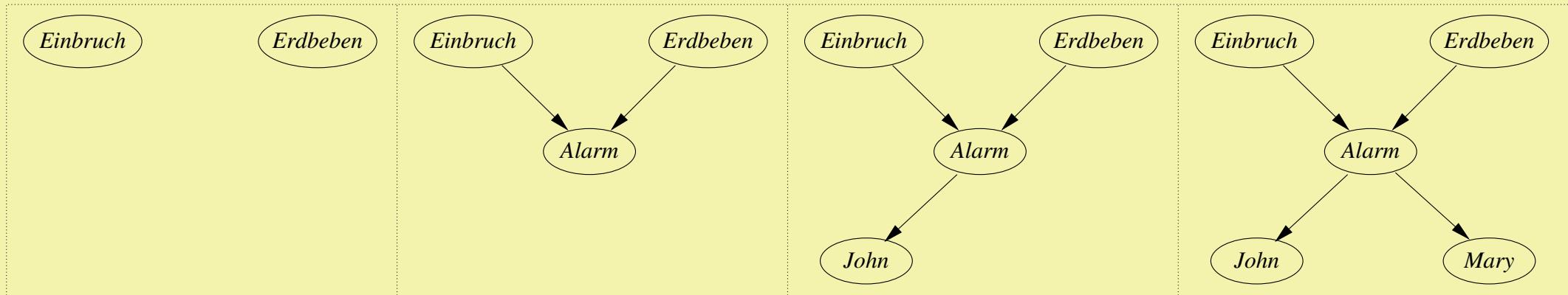
In der Reihenfolge (Leuko, TRek, Abw, Alt, Losl, Sono, Ersch, Darmg, Sex, S4Q, S1Q, S2Q, RektS, PathU, S3Q, Bef4) berechnet man

$$\sum_{i=1}^n (|v_i| - 1) \prod_{j=1}^{k_i} |e_{ij}| = 6 \cdot 6 \cdot 4 + 5 \cdot 4 + 2 \cdot 4 + 9 \cdot 7 \cdot 4 + 1 \cdot 3 \cdot 4 + 1 \cdot 4 + 1 \cdot 2 \cdot 4 \\ + 3 \cdot 3 \cdot 4 + 1 \cdot 4 + 1 \cdot 4 \cdot 2 + 1 \cdot 4 \cdot 2 + 1 \cdot 4 + 1 \cdot 4 + 1 \cdot 4 + 1 \cdot 4 + 1 = 521.$$

Kausalität und Netzstruktur

Aufbau eines Bayes-Netzes:

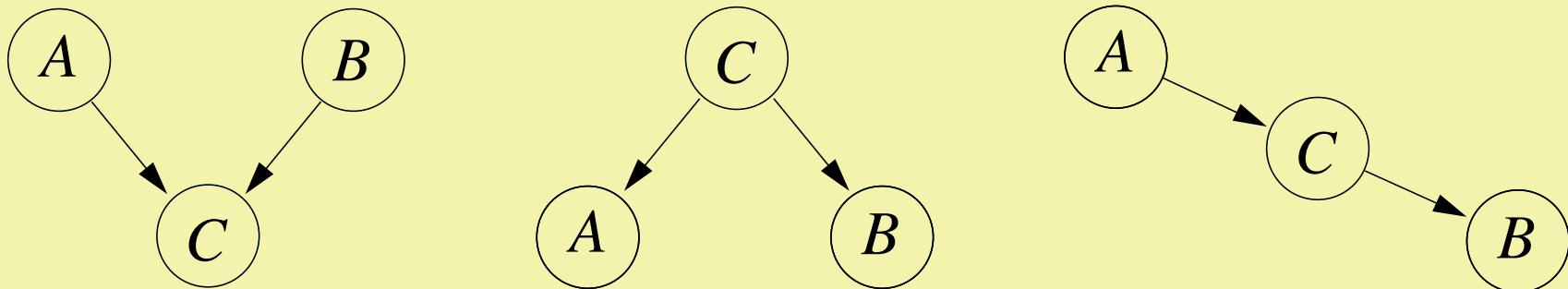
1. Entwurf der Netzwerkstruktur
meist manuell
 2. Eintragen der Wahrscheinlichkeiten in die CPTs
meist automatisch
- Ursachen *Einbruch* und *Erdbeben*
 - Symptome *John* und *Mary*
 - Alarm: nicht beobachtbare Variable
 - Kausalität beachten: von Ursache zu Wirkung vorgehen



Schrittweiser Aufbau des Alarm-Netzes unter Beachtung der Kausalität.

Vgl.: Aufgabe ??

Semantik von Bayes-Netzen



Zwischen zwei Knoten A und B wird keine Kante eingetragen, wenn sie unabhängig (links) oder bedingt unabhängig sind (Mitte, rechts).

Forderungen:

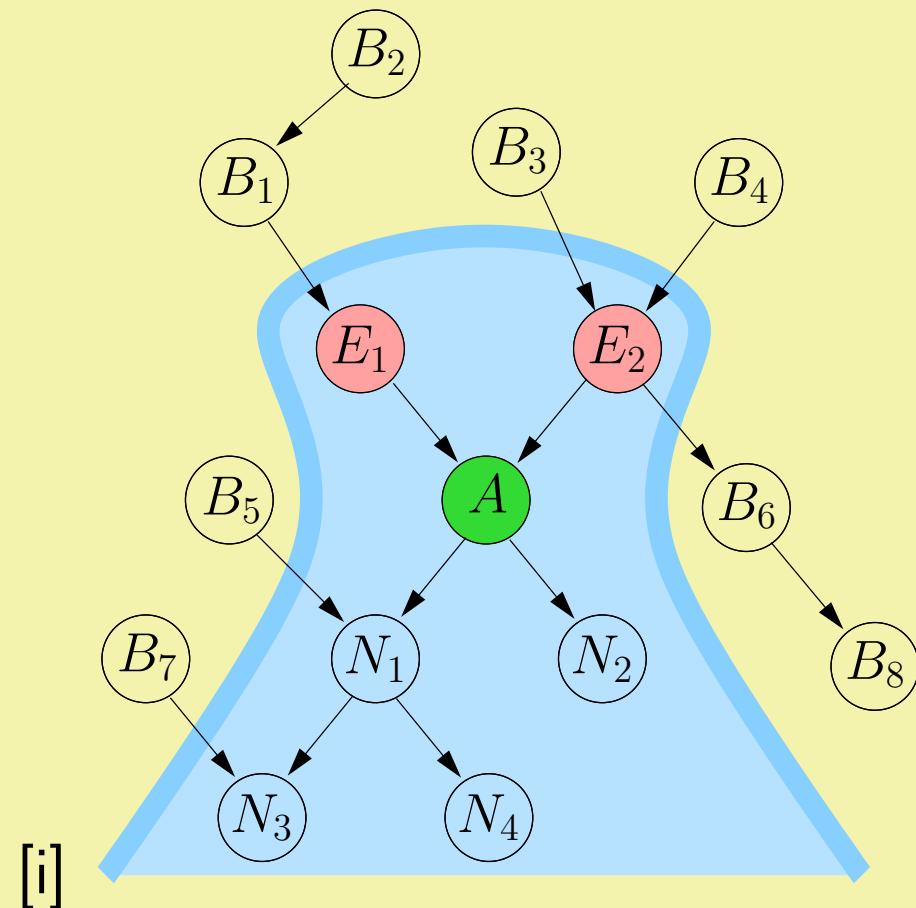
- Bayes-Netz hat keine Zyklen
- keine Variable hat einen Nachfolger mit kleinerer Nummer

Es gilt

$$\mathbf{P}(X_n | X_1, \dots, X_{n-1}) = \mathbf{P}(X_n | \text{Eltern}(X_n)).$$

Satz 7.12 Ein Knoten in einem Bayes-Netz ist bedingt unabhängig von allen Nicht-Nachfolgern, gegeben seine Eltern.

Sind die Elternknoten E_1 und E_2 gegeben, so sind alle Nichtnachfolger B_1, \dots, B_8 unabhängig von A .



Kettenregel für Bayes-Netze:

$$\mathbf{P}(X_1, \dots, X_n) = \prod_{i=1}^n \mathbf{P}(X_i | X_1, \dots, X_{i-1}) = \prod_{i=1}^n \mathbf{P}(X_i | \text{Eltern}(X_i))$$

Damit gilt Gleichung 7.22

$$\mathbf{P}(J, Ein, AI) = \mathbf{P}(J|AI)\mathbf{P}(AI|Ein)\mathbf{P}(Ein)$$

Die wichtigsten Begriffe und Grundlagen von Bayes-Netzen sind nun bekannt und wir können diese zusammenfassen Jensen:

Definition 7.14 Ein Bayes-Netz ist definiert durch:

- Eine Menge von Variablen und einer Menge von gerichteten Kanten zwischen diesen Variablen.
- Jede Variable hat endlich viele mögliche Werte.
- Die Variablen zusammen mit den Kanten stellen einen gerichteten azyklischen Graphen (engl. directed acyclic graph, DAG) dar. Ein DAG ist ein gerichteter Graph ohne Zyklen, das heißt ohne Pfade der Form (A, \dots, A) .
- Zu jeder Variablen A ist die CPT, das heißt die Tabelle der bedingten Wahrscheinlichkeiten $\mathbf{P}(A|Eltern(A))$, angegeben.

Zwei Variablen A und B heißen **bedingt unabhängig** gegeben C , wenn $\mathbf{P}(A, B|C) = \mathbf{P}(A|C) \cdot \mathbf{P}(B|C)$, bzw. wenn $\mathbf{P}(A|B, C) = \mathbf{P}(A|C)$.

Neben den grundlegenden Rechenregeln für Wahrscheinlichkeiten gelten folgende Regeln:

Bayes-Formel:
$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Marginalisierung:

$$P(B) = P(A, B) + P(\neg A, B) = P(B|A) \cdot P(A) + P(B|\neg A) \cdot P(\neg A)$$

Konditionierung: $P(A|B) = \sum_c P(A|B, C = c)P(C = c|B)$

Eine Variable in einem Bayes-Netz ist bedingt unabhängig von allen Nicht-Nachfolge-Variablen, gegeben ihre Eltern-Variablen. Wenn X_1, \dots, X_{n-1} keine Nachfolger von X_n sind, gilt $\mathbf{P}(X_n|X_1, \dots, X_{n-1}) = \mathbf{P}(X_n|\text{Eltern}(X_n))$. Diese Bedingung muss beim Aufbau eines Netzes beachtet werden.

Beim Aufbau eines Bayes-Netzes sollten die Variablen in Sinne der Kausalität angeordnet werden. Zuerst die Ursachen, dann die verdeckten Variablen und zuletzt die Diagnosevariablen.

Kettenregel: $\mathbf{P}(X_1, \dots, X_n) = \prod_{i=1}^n \mathbf{P}(X_i | \text{Eltern}(X_i))$

Siehe auch: d-Separation in [Pearl](#) und [Jensen](#).

Zusammenfassung

- Wahrscheinlichkeitslogik zum Schließen mit unsicherem Wissen
- Methode der maximalen Entropie
- modelliert nichtmonotones Schließen
- Bayes-Netze als ein Spezialfall von MaxEnt
- Bayes-Netze enthalten Unabhängigkeitsannahmen beim Aufbau eines Bayes-Netzes müssen alle CPTs ganz gefüllt werden
- Bei MaxEnt kann man beliebiges Wissen formulieren. Beispiel:
„Ich bin mir ziemlich sicher, dass A gilt.“: $P(A) \in [0.6, 1]$
- Inkonsistenz: $P(A) = 0.7$ und $P(A) = 0.8$
- PIT erkennt die Inkonsistenz
- In manchen Fällen trotzdem Schließen möglich

Kombination von MaxEnt und Bayes-Netzen

- Nach Bayessche Methodik ein Netz aufbauen
- Fehlende Werte für CPTs können durch Intervalle ersetzt werden
- oder durch andere Formeln der Wahrscheinlichkeitslogik.
- MaxEnt-System vervollständigt die Regelmenge

LEXMED

- medizinisches Expertensystem LEXMED
- besser als lineare Score-Systeme
- Scores sind äquivalent Spezialfall Naive-Bayes, das heißt zur Annahme, alle Symptome sind bedingt unabhängig gegeben die Diagnose (siehe Abschnitt 8.5, Aufgabe ??).
- LEXMED kann aus Daten einer Datenbank Wissen lernen (siehe Kapitel 8).

Ausblick

- Bayessche Schließen ist heute sehr wichtig und weit entwickelt
- Umgang mit stetigen Variablen bedingt möglich
- Kausalität in Bayes-Netzen
- ungerichtete Bayes-Netze
- Literatur: Pearl; Jensen; Whittaker; Duda/Hart/Stork
- Association for Uncertainty in Artificial Intelligence (AUAI) (www.auai.org).
- UAI-Konferenz

Kapitel 8

Maschinelles Lernen und Data Mining

Warum Maschinelles Lernen?

Elaine Rich¹:

„Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.“,

und:

Menschen lernen besser als Computer

⇒

Maschinelles Lernen ist für die KI sehr wichtig

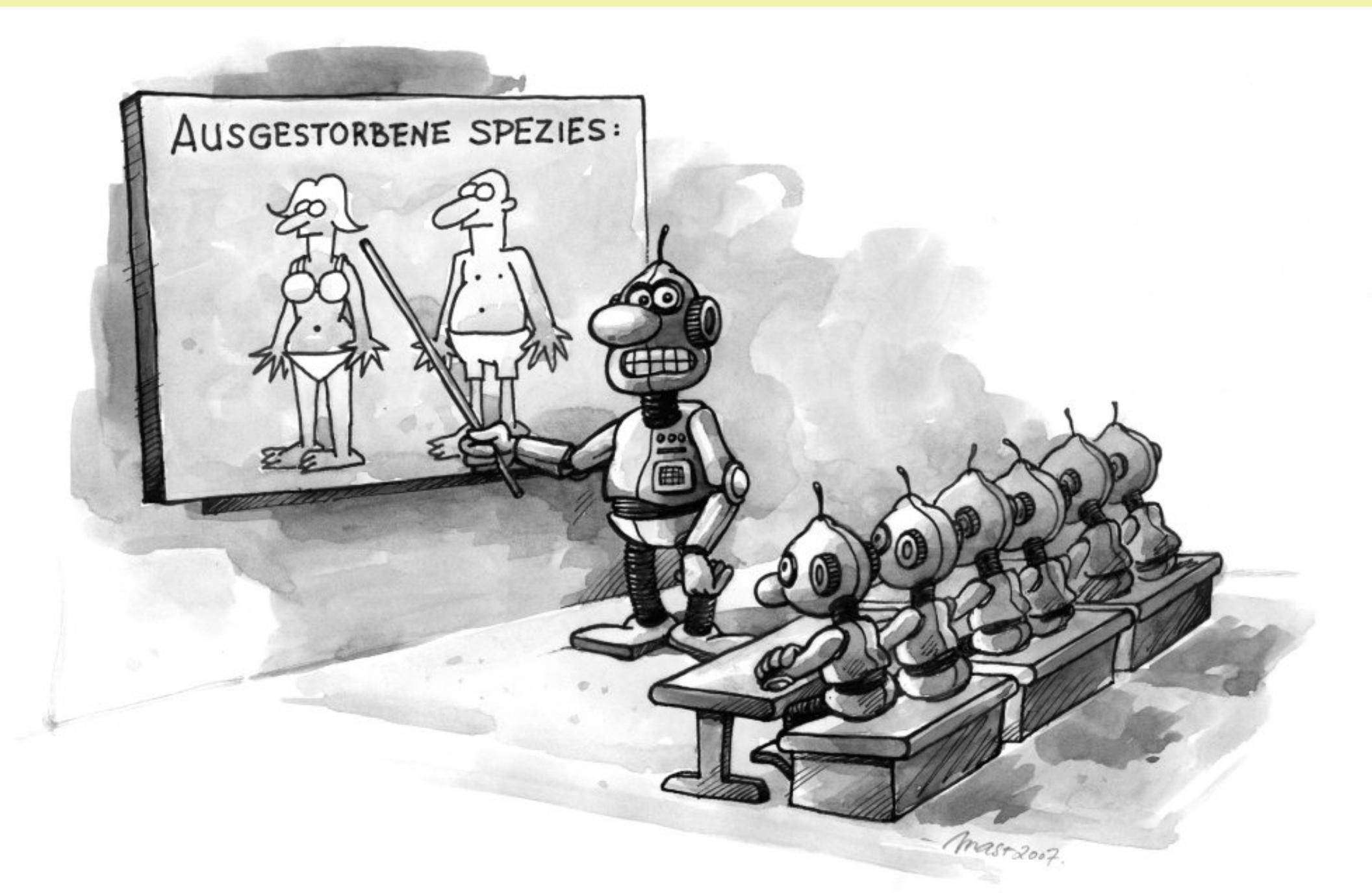
¹Rich, E. Artificial Intelligence. McGraw-Hill, 1983.

Warum Maschinelles Lernen?

- Komplexität der Software-Entwicklung, z.B.
 - Verhalten eines autonomen Roboters
 - Expertensysteme
 - Spamfilter
- Lösung: hybride Mischung aus programmiertem und gelerntem Komponenten.

Was ist Lernen?

- Lernen von Vokabeln einer Fremdsprache?
- Auswendiglernen eines Gedichts?
- Lernen mathematischer Fertigkeiten?
- Lernen des Skifahrens?



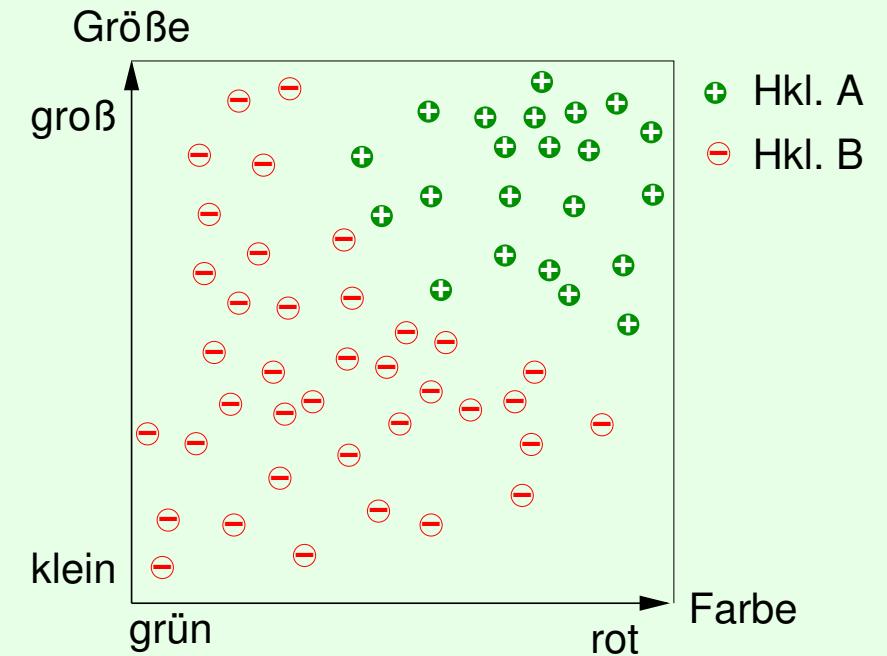
Beispiel: Sortieranlage für Äpfel

Merkmale (engl. **features**): Größe und Farbe

Klassifikaionsaufgabe: Äpfel in HKl. A, bzw. B einteilen (**Classifier**)

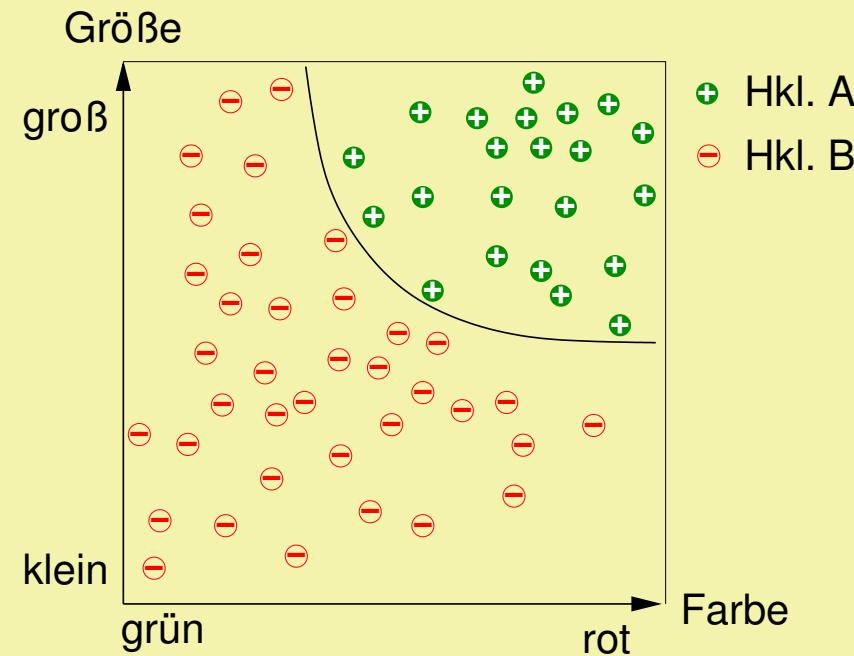
Größe [cm]	8	8	6	3	...
Farbe [0: grün, 1: rot]	0.1	0.3	0.9	0.8	...
Handelsklasse	B	A	A	B	...

Trainingsdaten für den Apfelsortieragenten.



Apfelsortieranlage und einige im Merkmalsraum klassifizierte Äpfel der Handelsklassen A und B.

Kurve trennt die Klassen



In der Praxis: 30 oder mehr Merkmale!

n Merkmale:

In n -dimensionalem **Merkmalsraum** ist eine $(n - 1)$ -dimensionale Hyperfläche zu finden, welche die beiden Klassen möglichst gut trennt.

Begriffe

Klassifikation / Classifier: bildet einen Merkmalsvektor auf einen Klassenwert ab. Feste Anzahl von Alternativen.

Klassifikation meist besser als Regression/Approximation

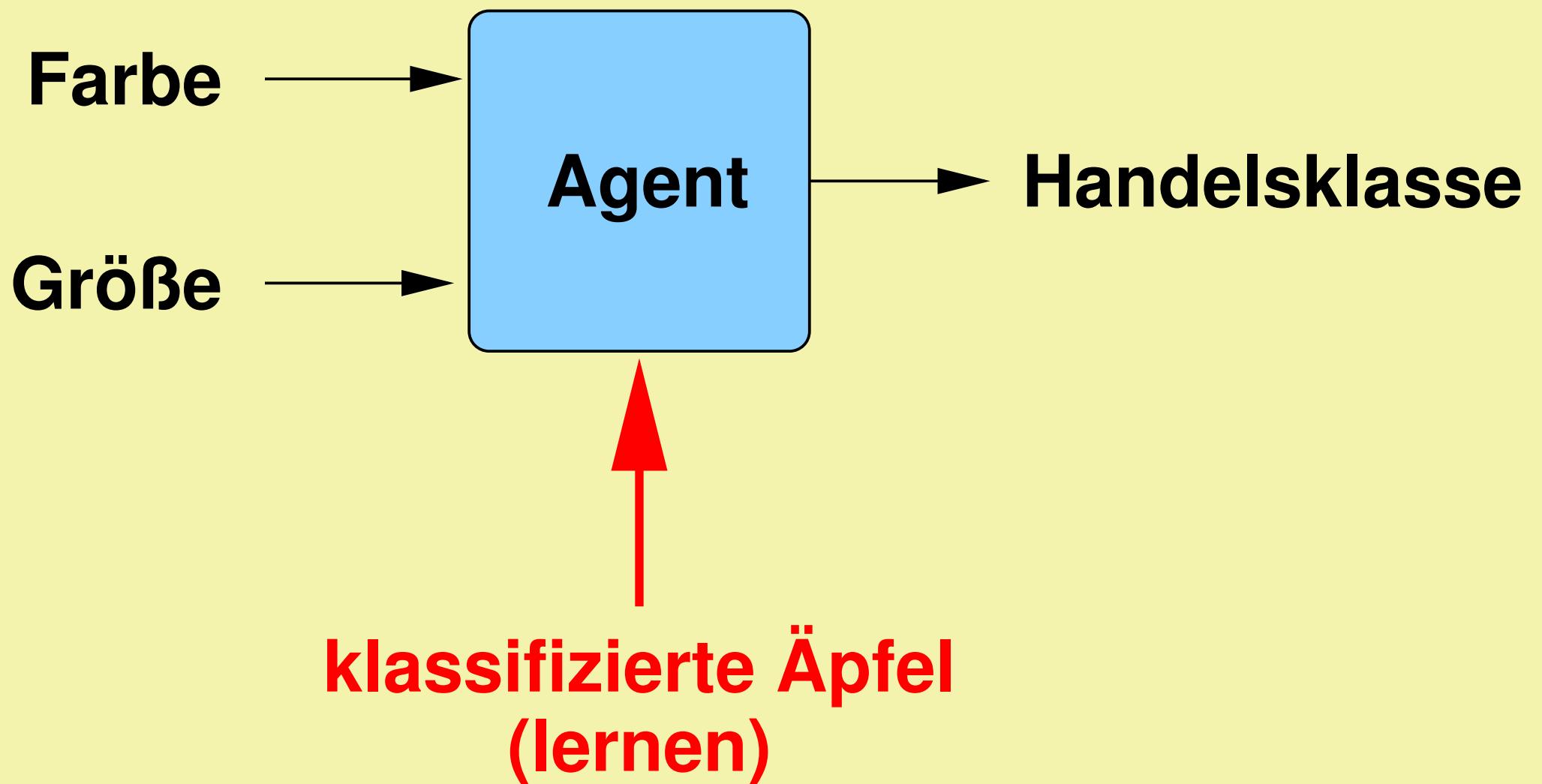
Beispiel: Apfelsortierung

Beispiel: Kurs Morgen > heute oder < heute vs. genauer Wert vorhersagen

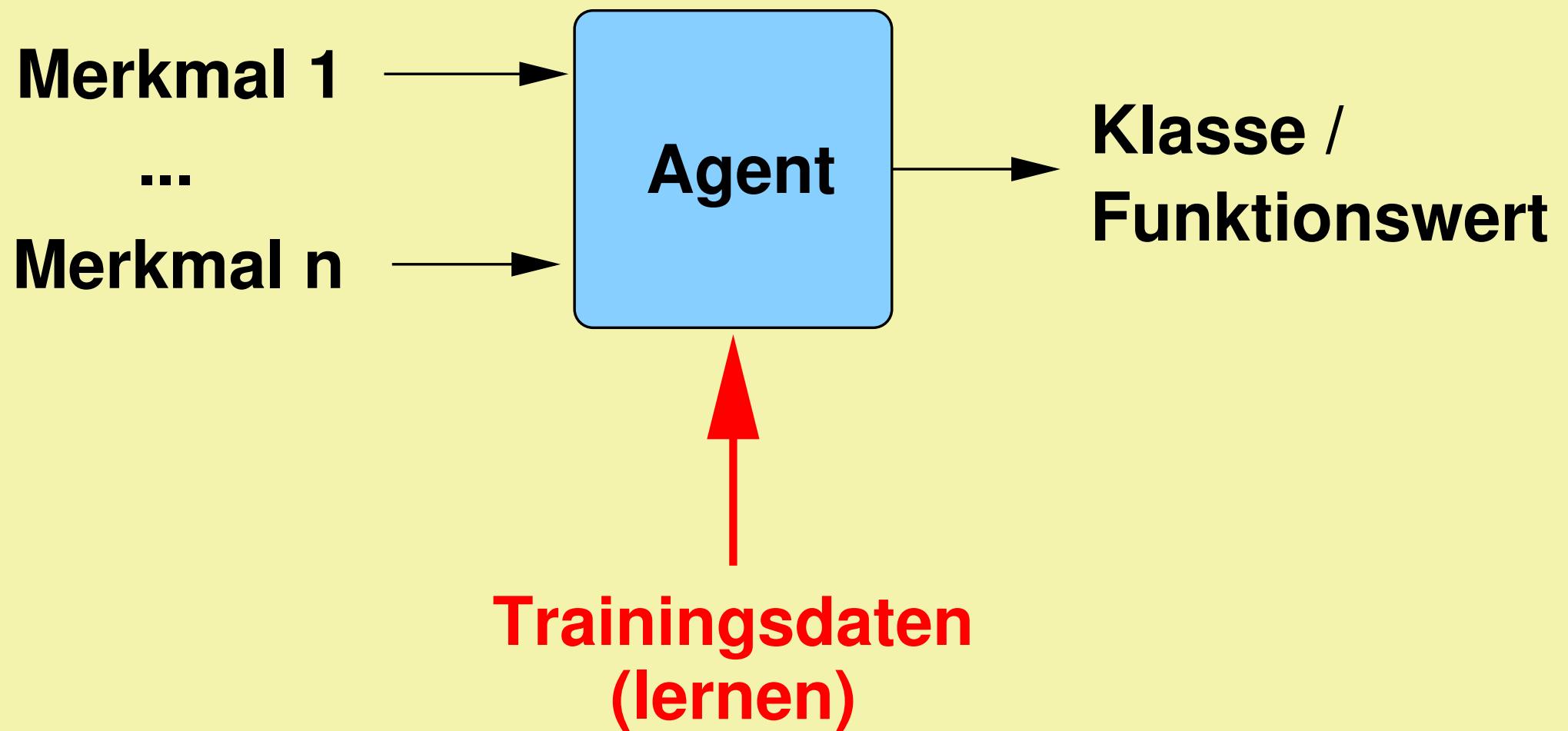
Approximation: bildet einen Merkmalsvektor auf eine reelle Zahl ab

Beispiel: Prognose des Kurses einer Aktie aus gegebenen Merkmalen

Der lernende Agent



Der lernende Agent, allgemein



Definition des Begriffs „Maschinelles Lernen“

2.

Machine Learning is the study of computer algorithms that improve automatically through experience.

In Anlehnung daran definieren wir nun:

Definition 8.2 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).

²Mitchell, T. Machine Learning. McGraw Hill, 1997.

Begriffe

am Beispiel der Apfelsortierung:

Aufgabe: Abbildung von Größe und Farbe eines Apfels auf die Handelsklasse lernen

Performance-Maß: Zahl der korrekt klassifizierten Äpfel

variabler Agent: (genauer eine Klasse von Agenten): das Lernverfahren bestimmt die Klasse aller möglichen Funktionen.

Trainingsdaten: (Erfahrung): Trainingsdaten enthalten das Wissen, welches von dem Lernverfahren extrahiert werden soll.

Testdaten: zeigen, ob der trainierte Agent gut von den gelernten auf neue Daten generalisieren kann.

Was ist Data-Mining?



Was ist Data-Mining?

- Wissen aus Daten extrahieren
- Wissen für Menschen verständlich machen
- Beispiel: Induktion von Entscheidungsbäumen
- Wissensmanagement:
 - Analyse von Kundenwünschen durch Statistik, z.B. in e-Shops
 - gezielte Werbung

Definition 8.4 *Der Prozess des Gewinnens von Wissen aus Daten sowie dessen Darstellung und Anwendung wird als **Data Mining** bezeichnet.*

Literatur: [Witten, I./Frank, E. Data Mining. Hanser Verlag München, 2001](#)

Datenanalyse

LEXMED-Daten, $N = 473$ Patienten:

Var.-Nr.	Beschreibung	Werte
1	Alter	stetig
2	Geschlecht (1=männl., 2=weibl.)	1,2
3	Schmerz Quadrant 1	0,1
4	Schmerz Quadrant 2	0,1
5	Schmerz Quadrant 3	0,1
6	Schmerz Quadrant 4	0,1
7	Lokale Abwehrspannung	0,1
8	Generalisierte Abwehrspannung	0,1
9	Schmerz bei Loslassmanoever	0,1
10	Erschuetterung	0,1
11	Schmerz bei rektaler Untersuchung	0,1
12	Temperatur axial	stetig
13	Temperatur rektal	stetig
14	Leukozyten	stetig
15	Diabetes mellitus	0,1
16	Appendizitis	0,1

hat Patient Bladdenerkrankung

Feature-Vektoren

$$\begin{aligned} \mathbf{x}^1 &= (26, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 37.9, 38.8, 23100, 0, 1) \\ \mathbf{x}^2 &= (17, 2, 0, 0, 1, 0, 1, 0, 1, 1, 0, 36.9, 37.4, 8100, 0, 0) \end{aligned}$$

Mittelwert

$$\bar{x}_i := \frac{1}{N} \sum_{p=1}^N x_i^p \rightarrow \text{notwendig für}$$

Standardabweichung

$$s_i := \sqrt{\frac{1}{N-1} \sum_{p=1}^N (x_i^p - \bar{x}_i)^2}.$$

Kovarianz

$$\sigma_{ij} = \frac{1}{N-1} \sum_{p=1}^N (x_i^p - \bar{x}_i)(x_j^p - \bar{x}_j)$$

Korrelationskoeffizient:

$[-1, +1]$

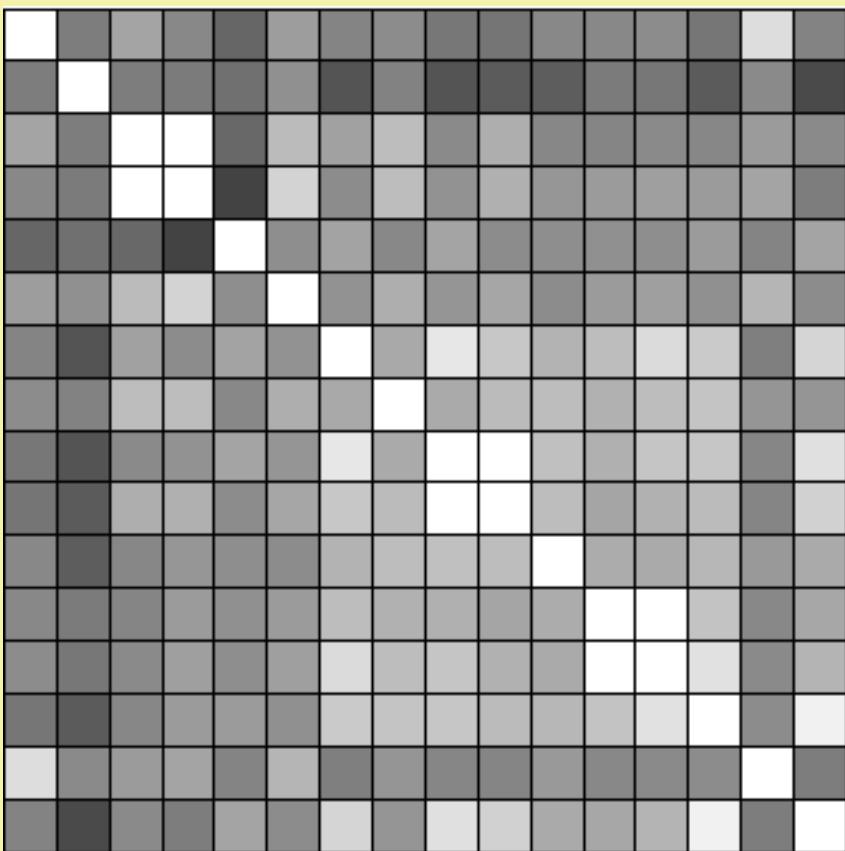
$$K_{ij} = \frac{\sigma_{ij}}{s_i \cdot s_j}.$$

1.	-0.009	0.14	0.037	-0.096	0.12	0.018	0.051	-0.034	0.041	0.034	0.037	0.05	-0.037	0.37	0.012
-0.009	1.	-0.0074	-0.019	-0.06	0.063	-0.17	0.0084	0.17	-0.14	-0.13	-0.017	0.034	0.14	0.045	-0.2
0.14	-0.0074	1.	0.55	-0.091	0.24	0.13	0.24	0.045	0.18	0.028	0.02	0.045	0.03	0.11	0.045
0.037	-0.019	0.55	1.	-0.24	0.33	0.051	0.25	0.074	0.19	0.087	0.11	0.12	0.11	0.14	-0.0091
-0.096	-0.06	-0.091	-0.24	1.	0.059	0.14	0.034	0.14	0.049	0.057	0.064	0.058	0.11	0.017	0.14
0.12	0.063	0.24	0.33	0.059	1.	0.071	0.19	0.086	0.15	0.048	0.11	0.12	0.063	0.21	0.053
0.018	-0.17	0.13	0.051	0.14	0.071	1.	0.16	0.4	0.28	0.2	0.24	0.36	0.29	-0.0001	0.33
0.051	0.0084	0.24	0.25	0.034	0.19	0.16	1.	0.17	0.23	0.24	0.19	0.24	0.27	0.083	0.084
-0.034	-0.17	0.045	0.074	0.14	0.086	0.4	0.17	1.	0.53	0.25	0.19	0.27	0.27	0.026	0.38
-0.041	-0.14	0.18	0.19	0.049	0.15	0.28	0.23	0.53	1.	0.24	0.15	0.19	0.23	0.02	0.32
0.034	-0.13	0.028	0.087	0.057	0.048	0.2	0.24	0.25	0.24	1.	0.17	0.17	0.22	0.098	0.17
0.037	-0.017	0.02	0.11	0.064	0.11	0.24	0.19	0.19	0.15	0.17	1.	0.72	0.26	0.035	0.15
0.05	-0.034	0.045	0.12	0.058	0.12	0.36	0.24	0.27	0.19	0.17	0.72	1.	0.38	0.044	0.21
-0.037	-0.14	0.03	0.11	0.11	0.063	0.29	0.27	0.27	0.23	0.22	0.26	0.38	1.	0.051	0.44
0.37	0.045	0.11	0.14	0.017	0.21	-0.0001	0.083	0.026	0.02	0.098	0.035	0.044	0.051	1.	-0.0055
0.012	-0.2	0.045	-0.0091	0.14	0.053	0.33	0.084	0.38	0.32	0.17	0.15	0.21	0.44	-0.0055	1.

Korrelationsmatrix für die 16 Appendizitis-Variablen, gemessen auf 473 Fällen.

Korrelationsmatrix als Dichteplot

useful

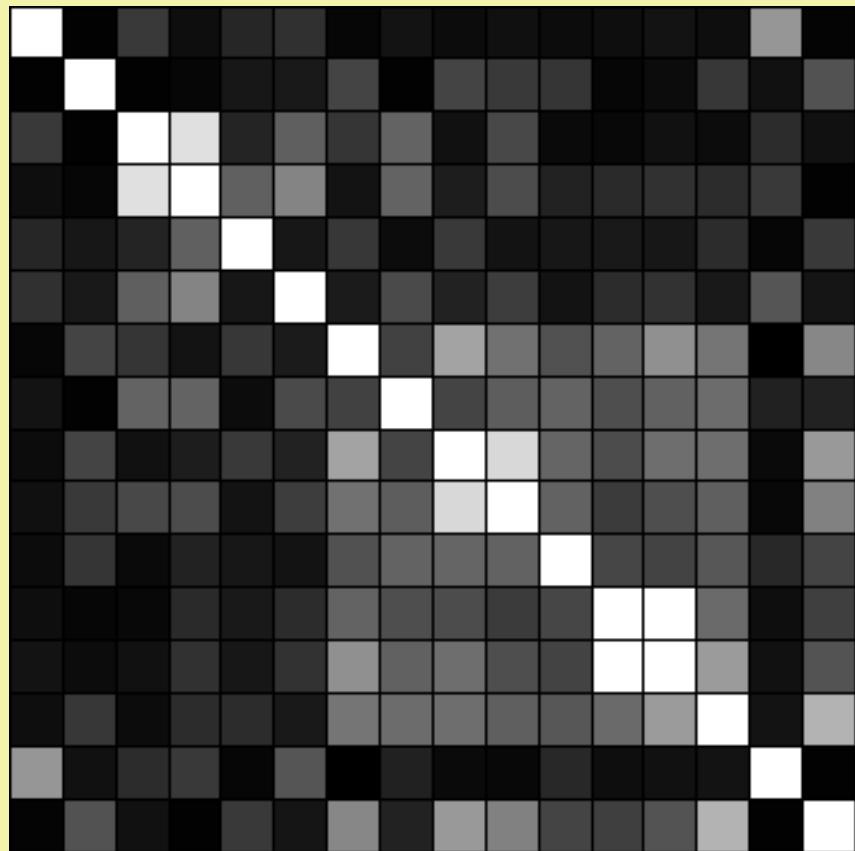
farbig \rightarrow Heatmap Skalar

$K_{ij} = -1$: schwarz, $K_{ij} = 1$: weiß

falls \rightarrow weiß ist: Korrelation nur da \rightarrow kein Machine Learning notwendig;
einfach den Daten LST anwenden

Aber: Kausalität nicht unbedingt vorhanden

Diagonale immer = 1

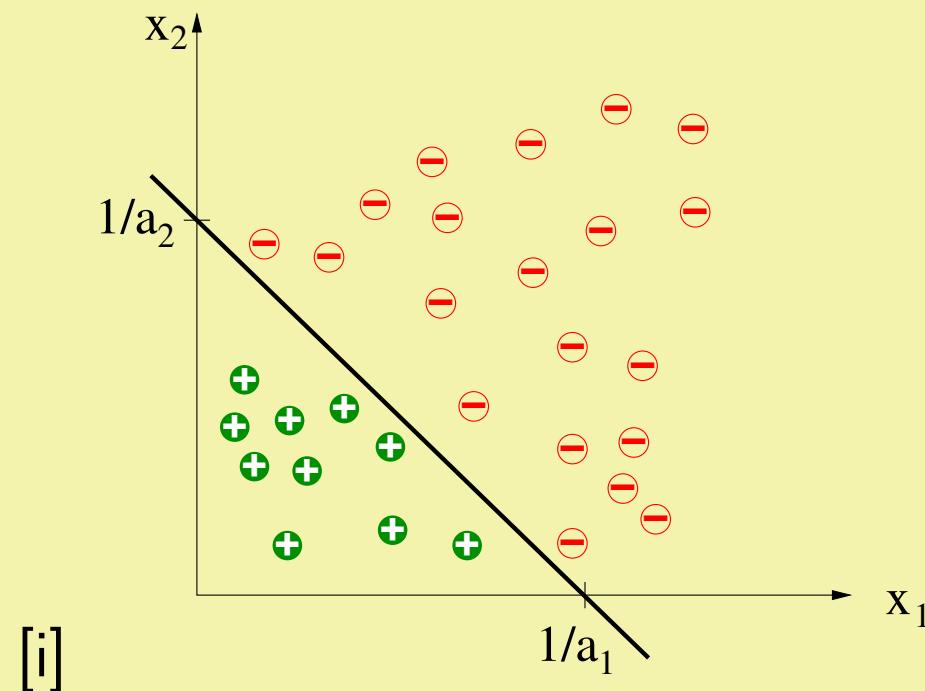


$|K_{ij}| = 0$: schwarz, $|K_{ij}| = 1$: weiß

Das Perzeptron, ein linearer Classifier

Eine linear separable zweidimensionale Datenmenge. Die Trenngerade hat die Gleichung

$$a_1x_1 + a_2x_2 = 1.$$



Lineare Separabilität

$n - 1$ -dimensionale Hyperebene im \mathbb{R}^n ist durch

$$\sum_{i=1}^n a_i x_i = \theta$$

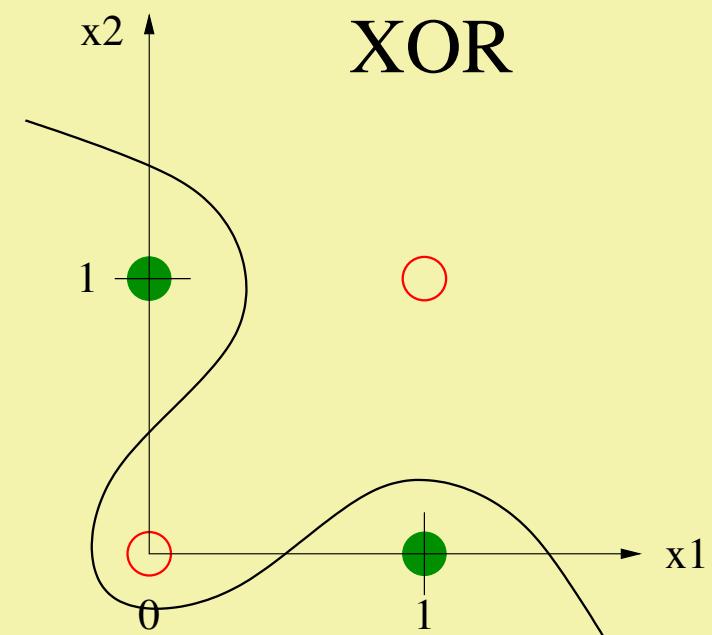
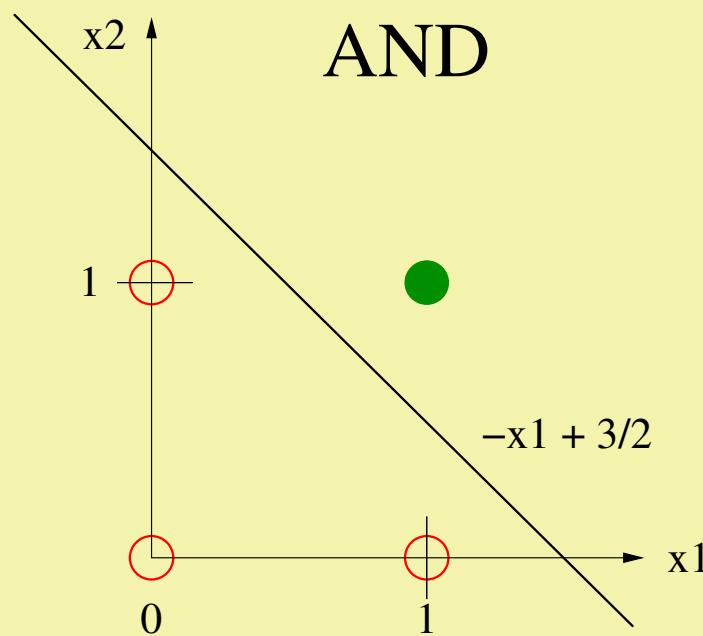
gegeben, also:

Definition 8.6 Zwei Mengen $M_1 \subset \mathbb{R}^n$ und $M_2 \subset \mathbb{R}^n$ heißen linear separabel, wenn reelle Zahlen a_1, \dots, a_n, θ existieren mit

$$\sum_{i=1}^n a_i x_i > \theta \quad \text{für alle } \mathbf{x} \in M_1 \quad \text{und} \quad \sum_{i=1}^n a_i x_i \leq \theta \quad \text{für alle } \mathbf{x} \in M_2.$$

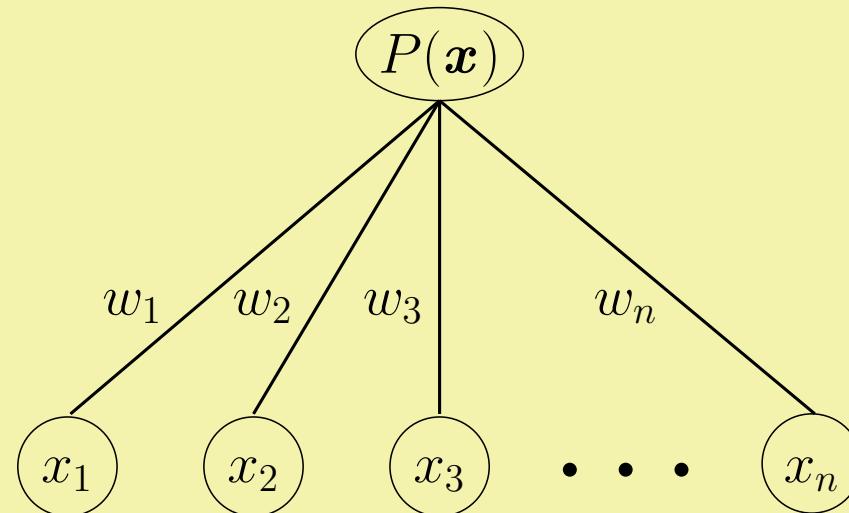
Der Wert θ wird als *Schwelle* bezeichnet.

Lineare Separabilität



AND ist linear separabel, XOR nicht. ($\bullet \hat{=} wahr$, $\circ \hat{=} falsch$).

Perzeptron



Definition 8.8 Sei $\mathbf{w} = (w_1, \dots, w_n) \in \mathbb{R}^n$ ein Gewichtsvektor und $\mathbf{x} \in \mathbb{R}^n$ ein Eingabevektor. Ein **Perzeptron** stellt eine Funktion $P : \mathbb{R}^n \rightarrow \{0, 1\}$ dar, die folgender Regel entspricht:

$$P(\mathbf{x}) = \begin{cases} 1 & \text{falls } \mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^n w_i x_i > 0 \\ 0 & \text{sonst} \end{cases}$$

Perzeptron

- zweilagiges gerichtetes Neuronales Netzwerk
- Die Eingabeveriablen x_i werden als **Merkmale** (engl. features) bezeichnet.
- trennende Hyperebene geht durch den Ursprung
- Punkte x über der Hyperebene $\sum_{i=1}^n w_i x_i = 0$ werden positiv ($P(x) = 1$) klassifiziert

Die Lernregel

M_+ : Menge der positiven Trainingsmuster

M_- : Menge der negativen Trainingsmuster

³, S. 164:

PERZEPTRONLERNEN(M_+, M_-)

w = beliebiger Vektor reeller Zahlen ungleich 0

Repeat

For all $x \in M_+$

If $w \cdot x \leq 0$ **Then** $w = w + x$

For all $x \in M_-$

If $w \cdot x > 0$ **Then** $w = w - x$

Until alle $x \in M_+ \cup M_-$ werden korrekt klassifiziert

³Minsky, M./Papert, S. Perceptrons. MIT Press, Cambridge, MA, 1969.

PERZEPTRONLERNEN(M_+, M_-)

...

For all $x \in M_+$

If $w \cdot x \leq 0$ **Then** $w = w + x$

For all $x \in M_-$

If $w \cdot x > 0$ **Then** $w = w - x$

...

$$(w + x) \cdot x = w \cdot x + x^2$$

$\Rightarrow w \cdot x$ wird irgend wann positiv!

$$(w - x) \cdot x = w \cdot x - x^2$$

$\Rightarrow w \cdot x$ wird irgend wann negativ!⁴

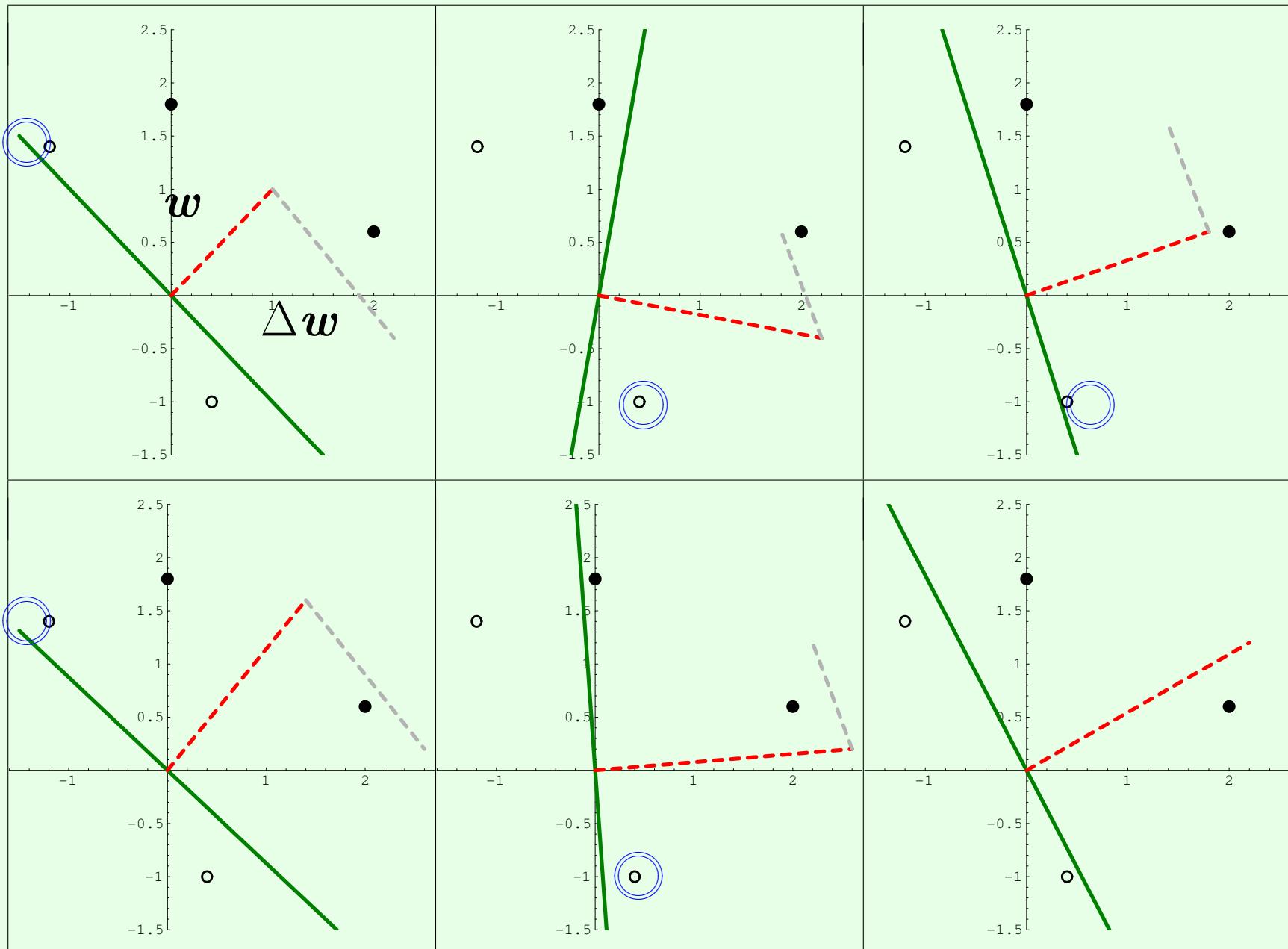
⁴Vorsicht! Dies ist kein Konvergenzbeweis!

Beispiel:

$$M_+ = \{(0, 1.8), (2, 0.6)\}, M_- = \{(-1.2, 1.4), (0.4, -1)\}$$

Initialer Gewichtsvektor: $w = (1, 1)$

Gerade $w \cdot x = x_1 + x_2 = 0$



$(-1.2, 1.4)$ falsch klassifiziert, denn $(-1.2, 1.4) \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0.2 > 0$.

Also $w = (1, 1) - (-1.2, 1.4) = (2.2, -0.4)$

Konvergenz

Satz 8.2 Es seien die Klassen M_+ und M_- linear separabel durch eine Hyperebene $\mathbf{w} \cdot \mathbf{x} = 0$. Dann konvergiert die PERZEPTRONLERNEN für jede Initialisierung ($\neq 0$) des Vektors w . Das Perzepton P mit dem so berechneten Gewichtsvektor trennt die Klassen M_+ und M_- , d.h.

$$P(\mathbf{x}) = 1 \iff \mathbf{x} \in M_+$$

und

$$P(\mathbf{x}) = 0 \iff \mathbf{x} \in M_-.$$

Lineare Separabilität

- Perzeptron kann nicht beliebige linear separable Mengen trennen
- im \mathbb{R}^2 Ursprungsgerade
- im \mathbb{R}^n Hyperebene im Ursprung,
- denn $\sum_{i=1}^n w_i x_i = 0$.

Trick

$$x_n := 1$$

Gewicht $w_n =: -\theta$ wirkt als Schwelle (**bias unit**), denn

$$\sum_{i=1}^n w_i x_i = \sum_{i=1}^{n-1} w_i x_i - \theta > 0 \Leftrightarrow \sum_{i=1}^{n-1} w_i x_i > \theta$$

- Anwendung: an jeden Trainingsdatenvektor ein 1-Bit anhängen!
- auch das Gewicht w_n , bzw. die Schwelle θ wird gelernt!

ein Perzeptron $P_\theta : \mathbb{R}^{n-1} \rightarrow \{0, 1\}$

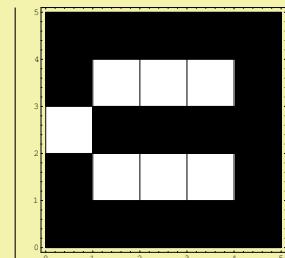
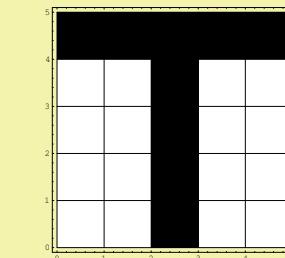
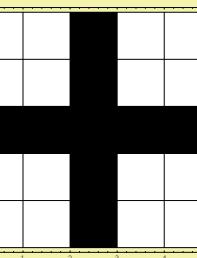
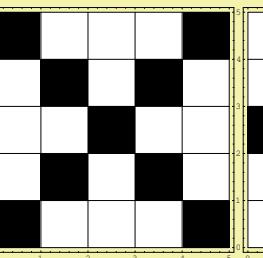
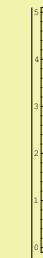
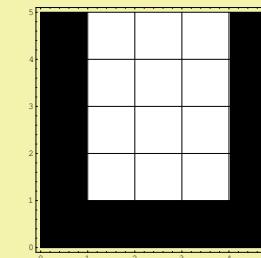
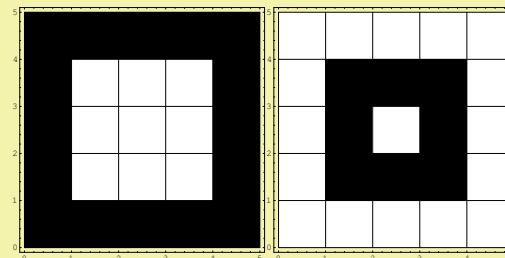
$$P_\theta(x_1, \dots, x_{n-1}) = \begin{cases} 1 & \text{falls } \sum_{i=1}^{n-1} w_i x_i > \theta \\ 0 & \text{sonst} \end{cases} \quad (8.1)$$

mit beliebiger Schwelle kann durch ein Perzepron $P : \mathbb{R}^n \rightarrow \{0, 1\}$ mit Schwelle 0 simuliert werden. Also:

Satz 8.4 Eine Funktion $f : \mathbb{R}^n \rightarrow \{0, 1\}$ kann von einem Perzepron genau dann dargestellt werden, wenn die beiden Mengen der positiven und negativen Eingabevektoren linear separabel sind.

Beweis: Vergleiche Gleichung 8.1 mit der Definition von linear separabel.

Beispiel:



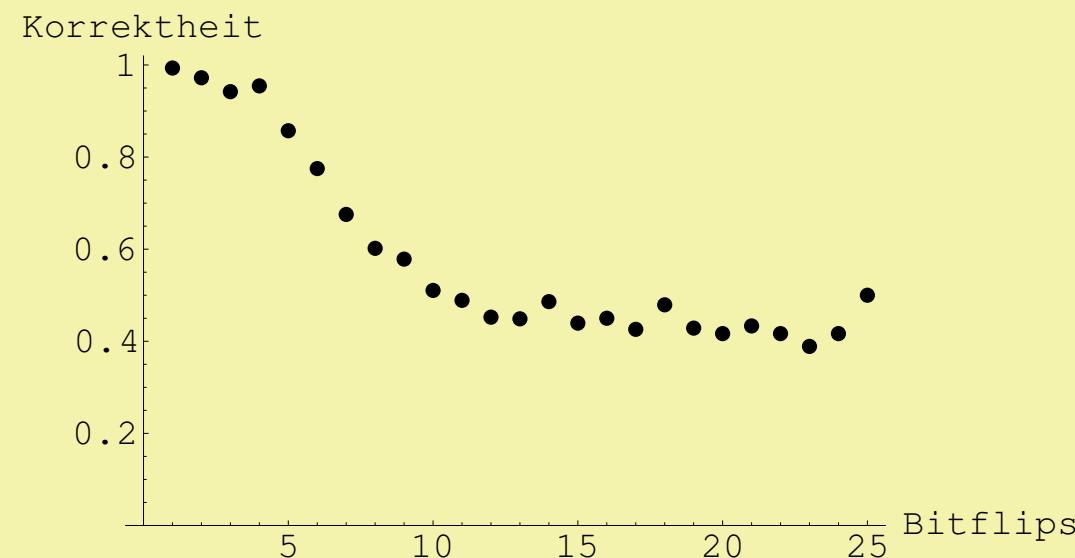
Positive Trainingsbeispiele (M_+)

Negative Trainingsbeispiele (M_-)

Testmuster

- Trainingsdaten werden von in 4 Iterationen über alle Muster gelernt
- Testen der Generalisierungsfähigkeit: verrauschte Muster mit einer variablen Anzahl invertierter (aufeinanderfolgender) Bits.

Relative Korrektheit des Perzeptrons in Abhängigkeit von der Zahl invertierter Bits in den Testdaten.



Optimierung und Ausblick

- langsame Konvergenz
- Beschleunigung durch Normierung der Gewichtsänderungsvektoren: $w = w \pm x / |x|$.
- Bessere Initialisierung des Vektors w .

$$w_0 = \sum_{x \in M_+} x - \sum_{x \in M_-} x,$$

die in Aufgabe ?? genauer untersucht werden soll.

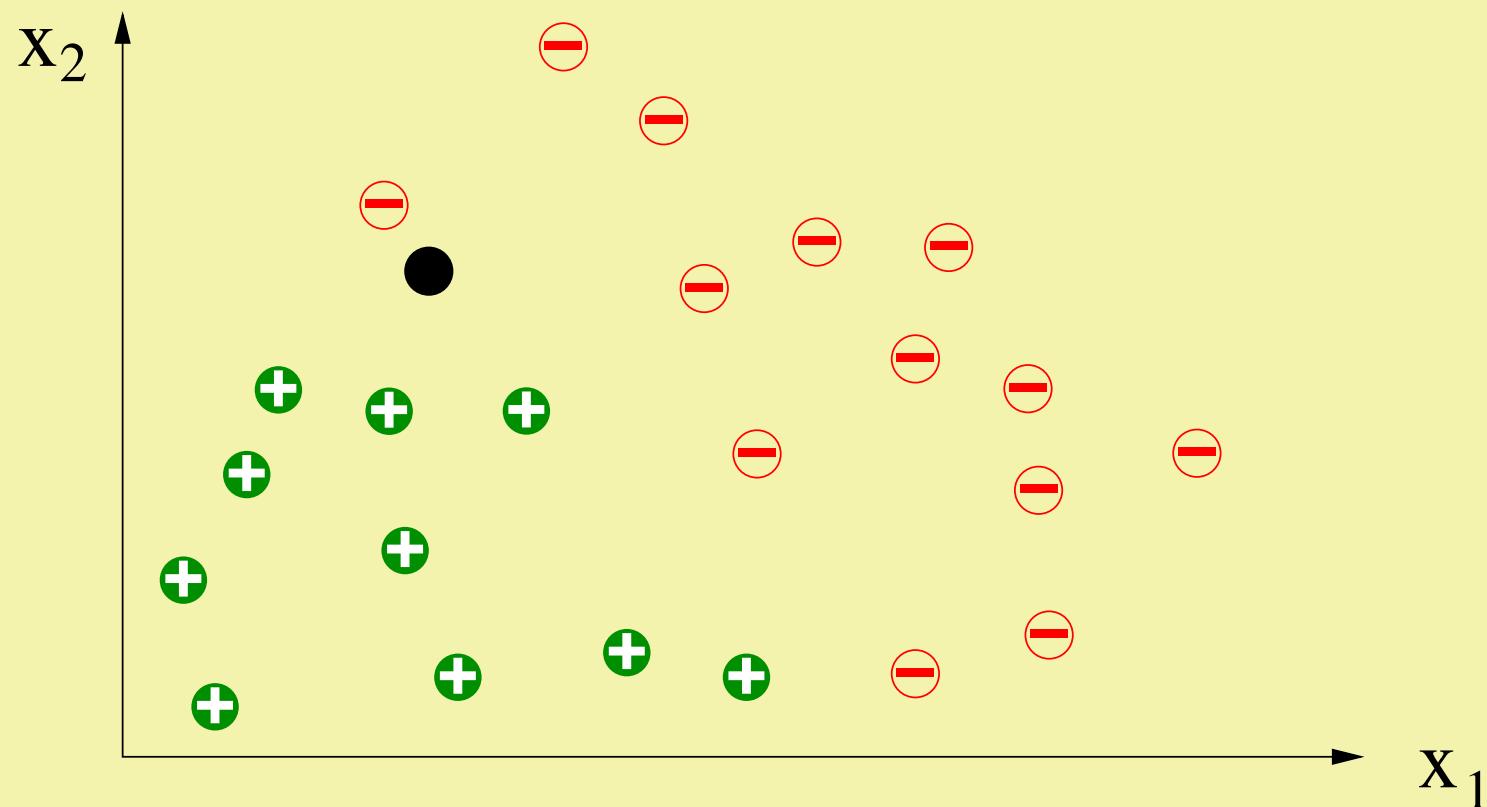
- mehrlagige Netze, z.B. Backpropagation sind mächtiger (Abschnitt ??)
- Backpropagation kann nicht linear separable Mengen trennen

Die Nearest Neighbour Methode

- Auswendiglernen mit Generalisierung
- Arzt merkt sich Fälle
- 1. bei der Diagnose: erinnern an ähnlich gelagerte Fälle aus der Vergangenheit.
- 2. gleiche oder eine ähnliche Diagnose stellen.
- Schwierigkeiten:
 - Arzt muss gutes Gefühl für **Ähnlichkeit** besitzen
 - Ist der gefundene Fall ähnlich genug?

Was heißt Ähnlichkeit?

Zwei Beispiele sind umso ähnlicher, je geringer ihr Abstand im Merkmalsraum ist.



Schwarz markierter Punkt wird negativ klassifiziert.

Abstand

Abstand $d(\mathbf{x}, \mathbf{y})$ zwischen zwei Punkten:

durch die euklidische Norm gegebenen Metrik

$$d(\mathbf{x}, \mathbf{y}) = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}.$$

gewichtete Merkmale:

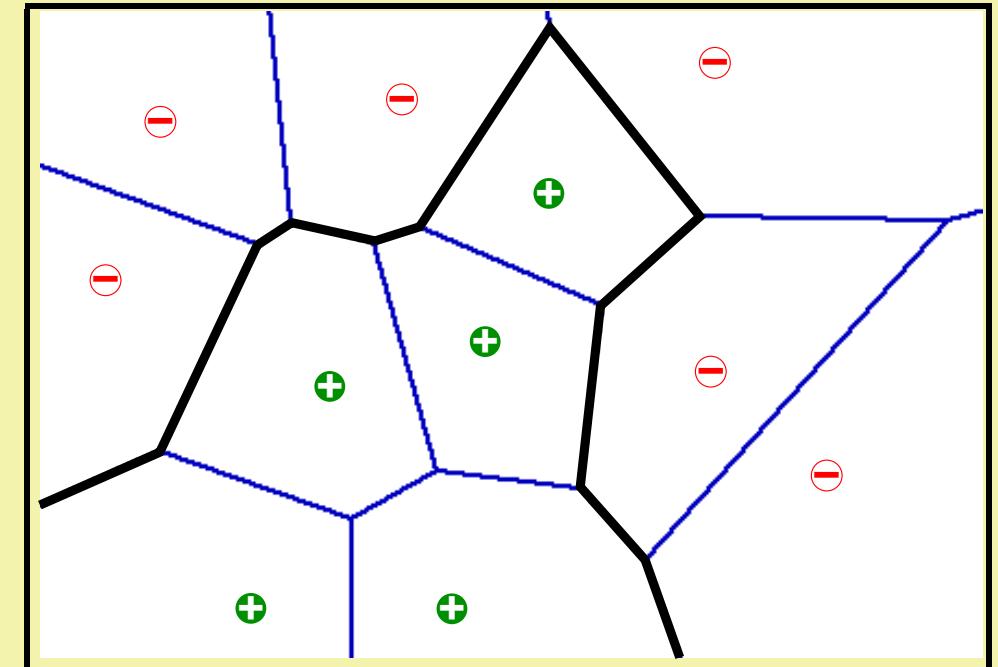
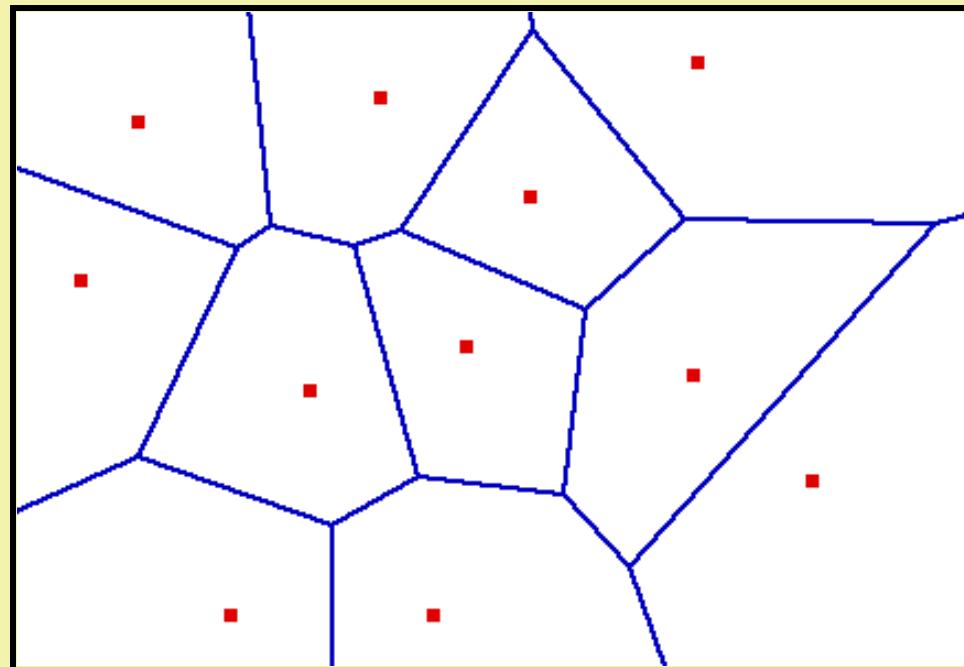
$$d_w(\mathbf{x}, \mathbf{y}) = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^n w_i(x_i - y_i)^2}.$$

Algorithmus

NEARESTNEIGHBOUR(M_+, M_-, s)

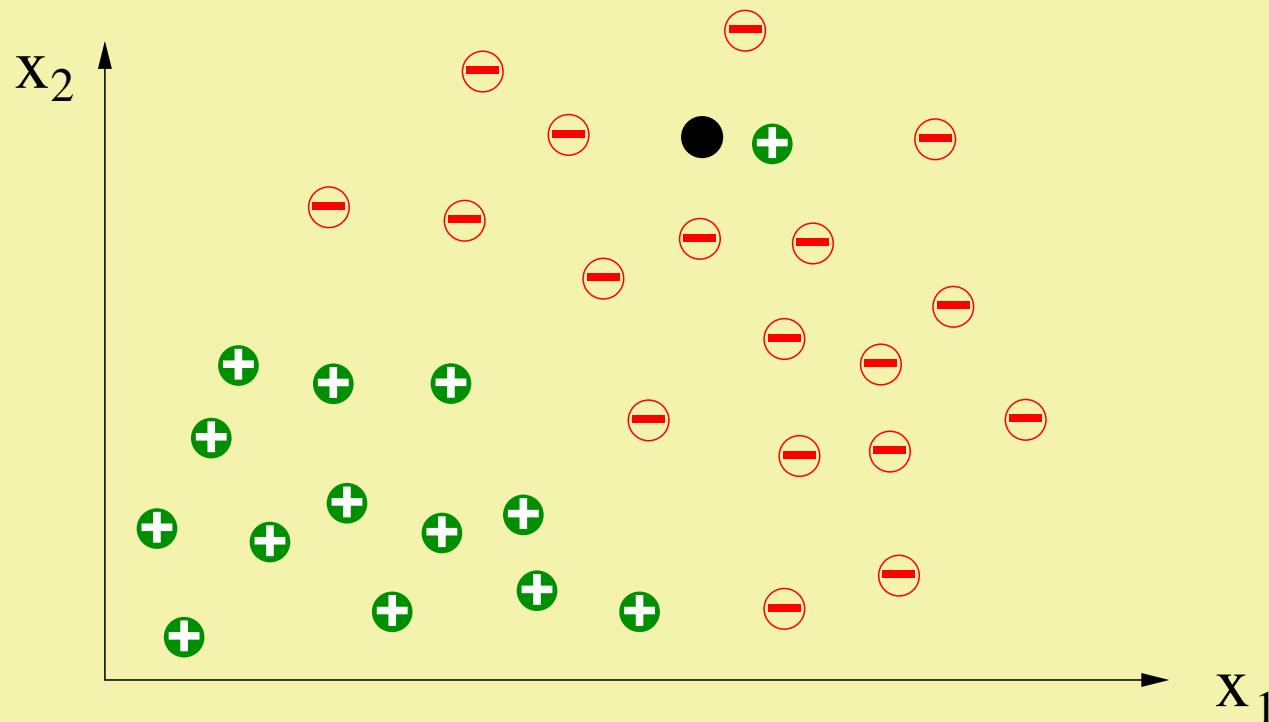
$t = \operatorname{argmin}_{x \in M_+ \cup M_-} \{d(s, x)\}$
If $t \in M_+$ **Then Return**(“+”)
Else Return(“-”)

Klassentrennung (virtuell)



Voronoidiagramm (links) und Klassentrennlinie (rechts).

- Nearest-Neighbour-Methode ist mächtiger als das Perceptron
- beliebig komplexe Trennlinien (Hyperflächen) möglich
- Gefahr durch Ausreißer
- Fehlanpassungen an zufällige Fehler (Rauschen)
- **Überanpassung** (engl. overfitting).



Schwarz markierter Punkt wird falsch klassifiziert.

k-Nearest-Neighbour-Methode

Mehrheitsentscheid unter den k nächsten Nachbarn:

Der Algorithmus K-NEARESTNEIGHBOUR.

K-NEARESTNEIGHBOUR(M_+, M_-, s)

$V = \{k \text{ nächste Nachbarn in } M_+ \cup M_-\}$

If $|M_+ \cap V| > |M_- \cap V|$ **Then** **Return**(“+”)

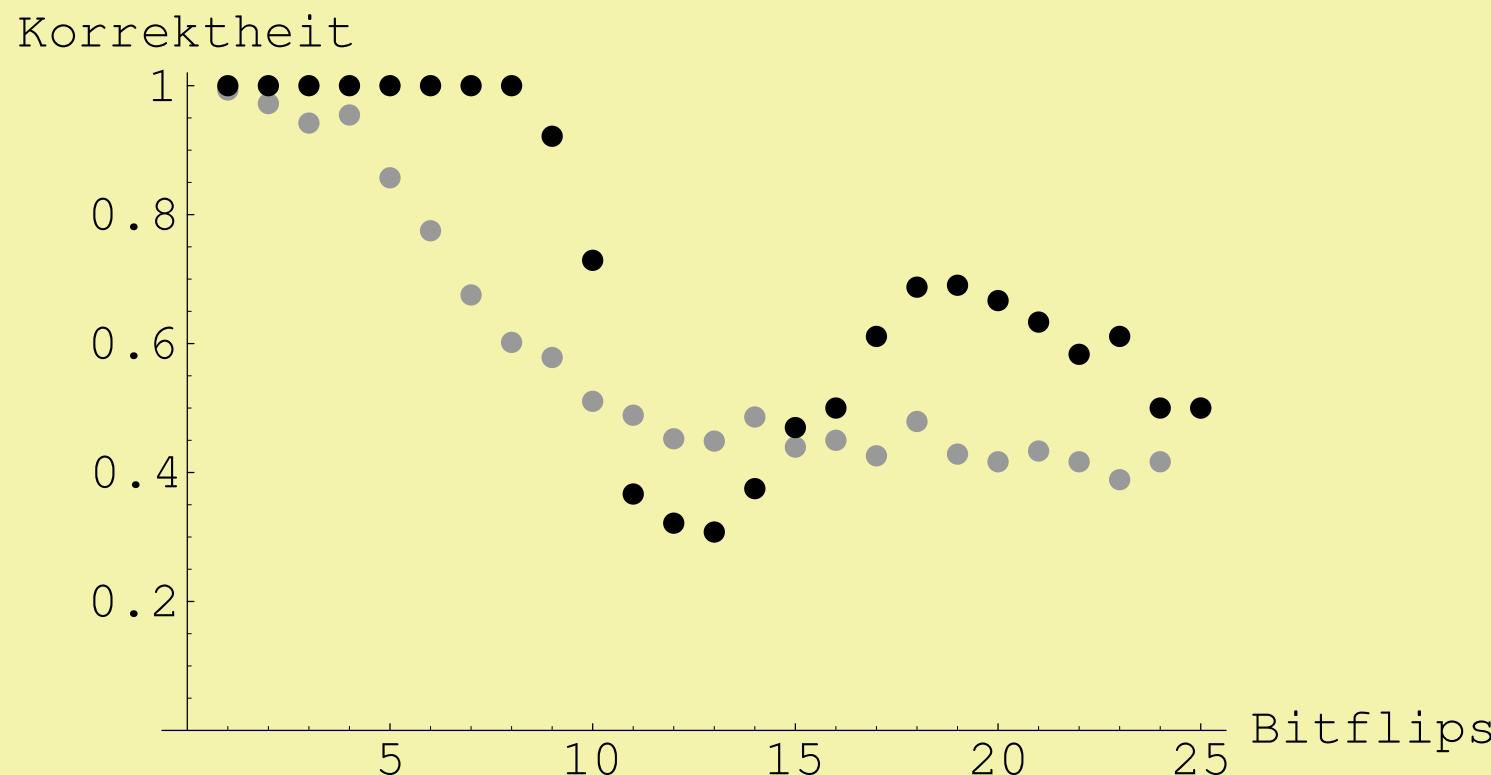
ElseIf $|M_+ \cap V| < |M_- \cap V|$ **Then** **Return**(“-”)

Else **Return**(Random(“,+”, “-”))

Beispiel:

- Wir wenden NEARESTNEIGHBOUR auf Beispiel 8.1 an.
- Hamming-Abstand als Metrik⁷
- Testbeispiele mit invertierten Bits wie beim Perzeptron

⁷Der Hamming Abstand zweier Bit-Vektoren ist die Anzahl unterschiedlicher Bits der beiden Vektoren.

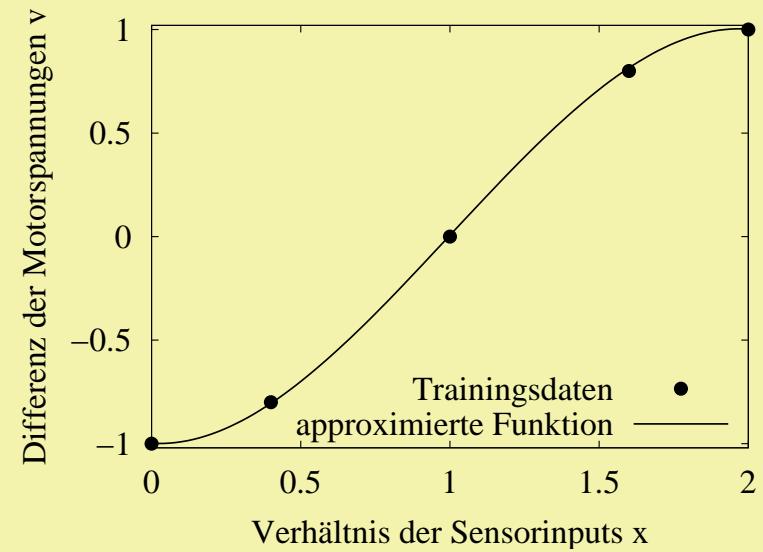
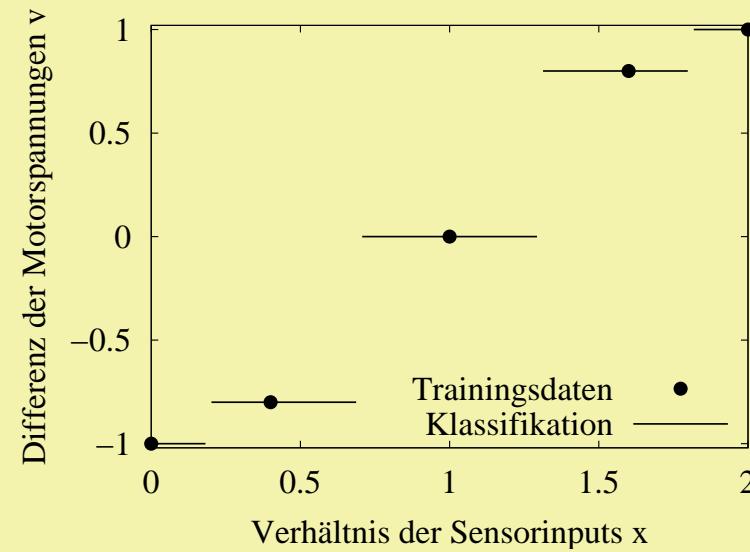
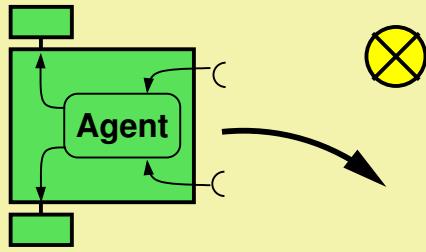


Korrekttheit der Nearest-Neighbour-Klassifikation (grau: Perzepron)

Mehr als zwei Klassen

- Nearest-Neighbour-Klassifikation für mehr als zwei Klassen:
Klasse des nächsten Nachbarn
- k-Nearest-Neighbour-Methode: Klasse mit Mehrheit unter den k nächsten Nachbarn.
- k-Nearest-Neighbour-Methode nur bei wenigen Klassen, bis etwa 10 Klassen

Beispiel: Ein autonomer Roboter mit einfacher Sensorik soll lernen, sich vom Licht wegzubewegen.



Sensorsignale s_l des linken und s_r des rechten Sensors, $x = s_r/s_l$

Aus x soll die Differenz $v = U_r - U_l$ der beiden Spannungen U_r am rechten und U_l am linken Motor bestimmt werden.

Der Agent muss eine Abbildung f finden, die für jeden Wert x den "richtigen" Wert $v = f(x)$ berechnet.

Approximationsmethoden

- Polynominterpolation, Spline-Interpolation, Methode der kleinsten Quadrate
- Problematisch in hohen Dimensionen.
- Schwierigkeit: in der KI werden modellfreie Approximationsmethoden benötigt
- Neuronale Netze

k-NN für Approximationsprobleme

1. Bestimmung der Menge $V = \{x_1, x_2, \dots, x_k\}$ der k nächsten Nachbarn von x
2. mittlerer Funktionswert

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^k f(x_i) \quad (8.2)$$

Je größer k wird, desto glatter wird die Funktion \hat{f} .

Der Abstand ist relevant

bei großem k :

- viele Nachbarn mit großem Abstand
- wenige Nachbarn mit kleinem Abstand.
- Berechnung von \hat{f} durch weit entfernte Nachbarn dominiert

Lösung: in K-NEARESTNEIGHBOUR werden “Stimmen” gewichtet mit

$$w_i = \frac{1}{1 + \alpha d(\mathbf{x}, \mathbf{x}_i)^2}, \quad (8.3)$$

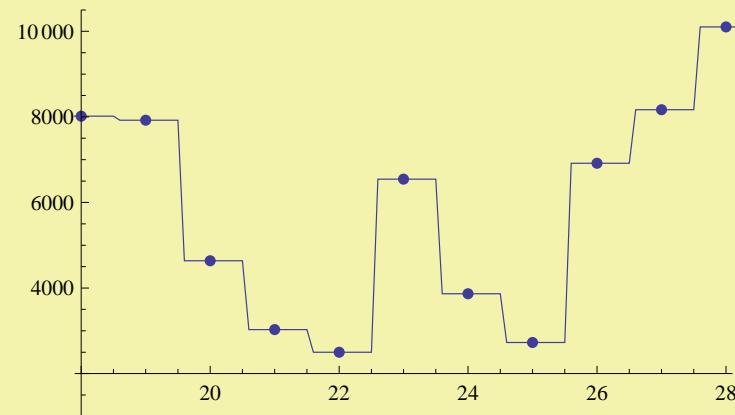
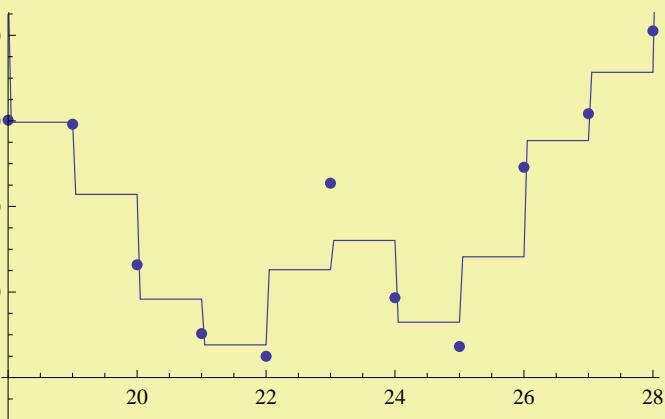
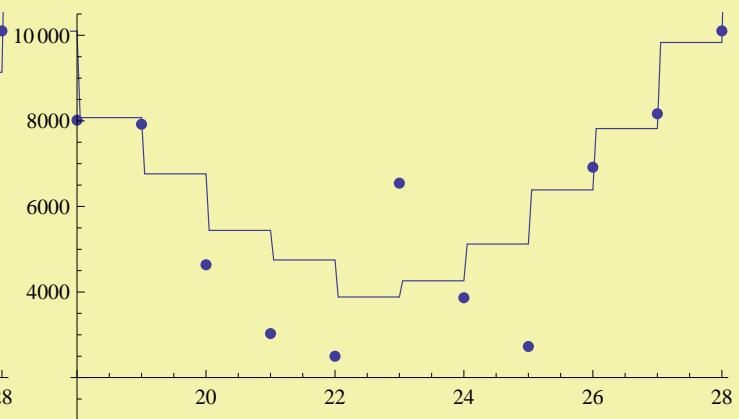
Approximation: Gleichung 8.2 wird ersetzt durch

$$\hat{f}(\mathbf{x}) = \frac{\sum_{i=1}^k w_i f(\mathbf{x}_i)}{\sum_{i=1}^k w_i}.$$

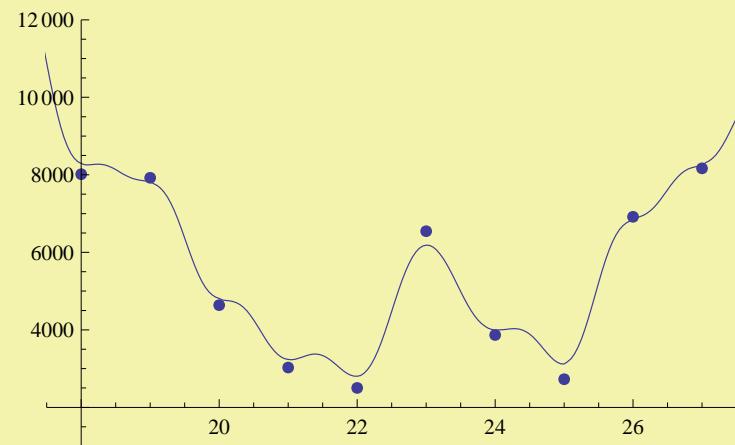
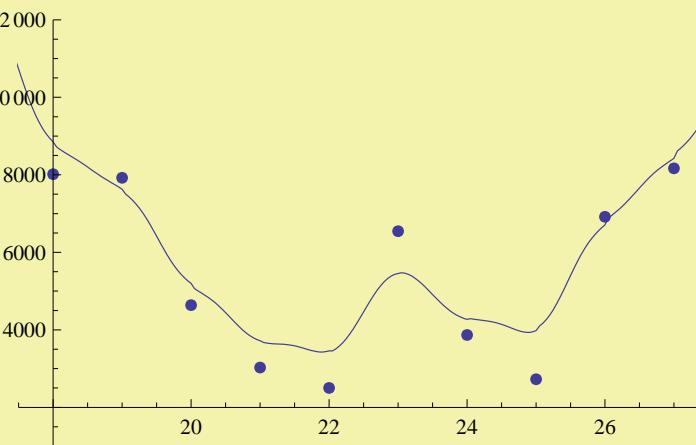
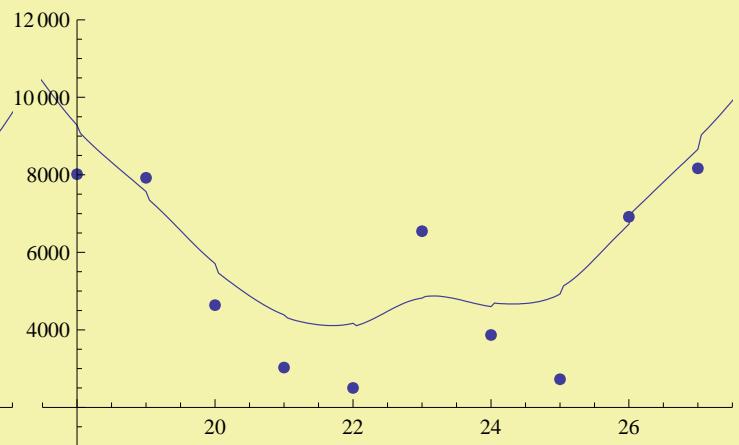
- Einfluß der Punkte für größer werdenden Abstand geht gegen Null.
- alle Trainingsdaten können zur Klassif./Approx. verwendet werden!

k-NN mit/ohne Abstandsgewichtung

k-Nearest Neighbour


 $k = 1$

 $k = 2$

 $k = 6$

Nearest Neighbour mit Abstandsgewichtung


 $\alpha = 20$

 $\alpha = 4$

 $\alpha = 1$

Rechenzeiten

- Lernen durch einfaches Abspeichern
- Daher sehr schnelles Lernen
- Klassifikation/Approximation eines Vektors x sehr aufwändig

Finden der k nächsten Nachbarn bei n Trainingsdaten: $\Theta(n)$

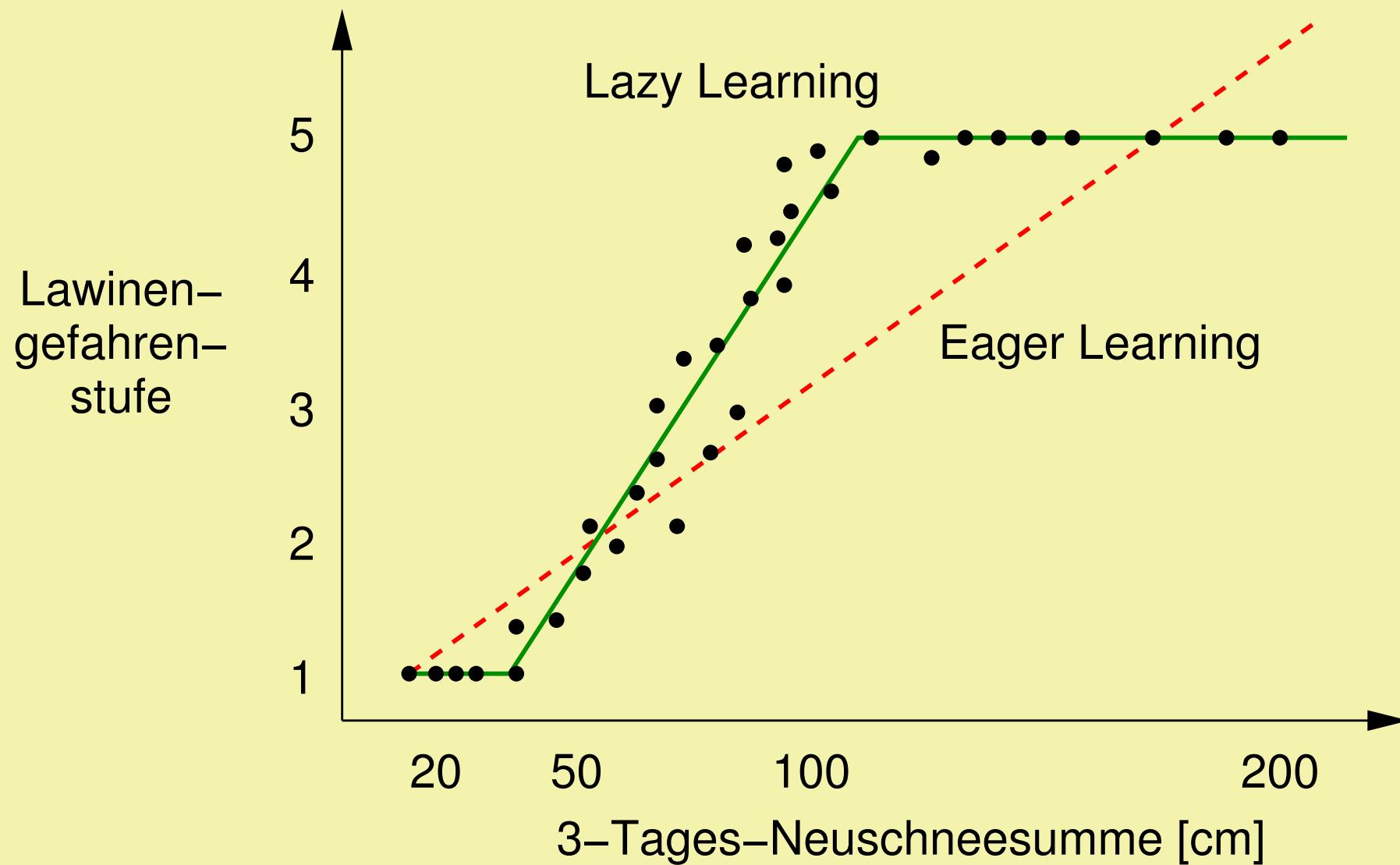
Klassifikation/Approximatio: $\Theta(k)$

Gesamtrechenzeit: $\Theta(n + k)$

Nearest-Neighbour-Methoden = **Lazy Learning**

Beispiel: Lawinenprognose

Aufgabe: aus der Neuschneemenge ist die aktuelle Lawinengefahr zu bestimmen.



NN-Methoden: Einsatzgebiete

Nearest-Neighbour-Methoden sind gut geeignet,

wenn eine gute lokale Approximation benötigt wird, die aber keine hohen Anforderungen an die Geschwindigkeit des Systems stellen.

Nearest-Neighbour-Methoden sind schlecht geeignet,

wenn eine für Menschen verständliche Beschreibung des in den Daten enthaltenen Wissens gefordert wird (Data Mining).

Fallbasiertes Schließen

((engl. case-based reasoning), kurz CBR):

- Erweiterung von NN-Methoden auf symbolische Problembeschreibungen und deren Lösung⁸.
- Einsatzgebiete:
 - Diagnose technischer Probleme
 - Telefonhotlines

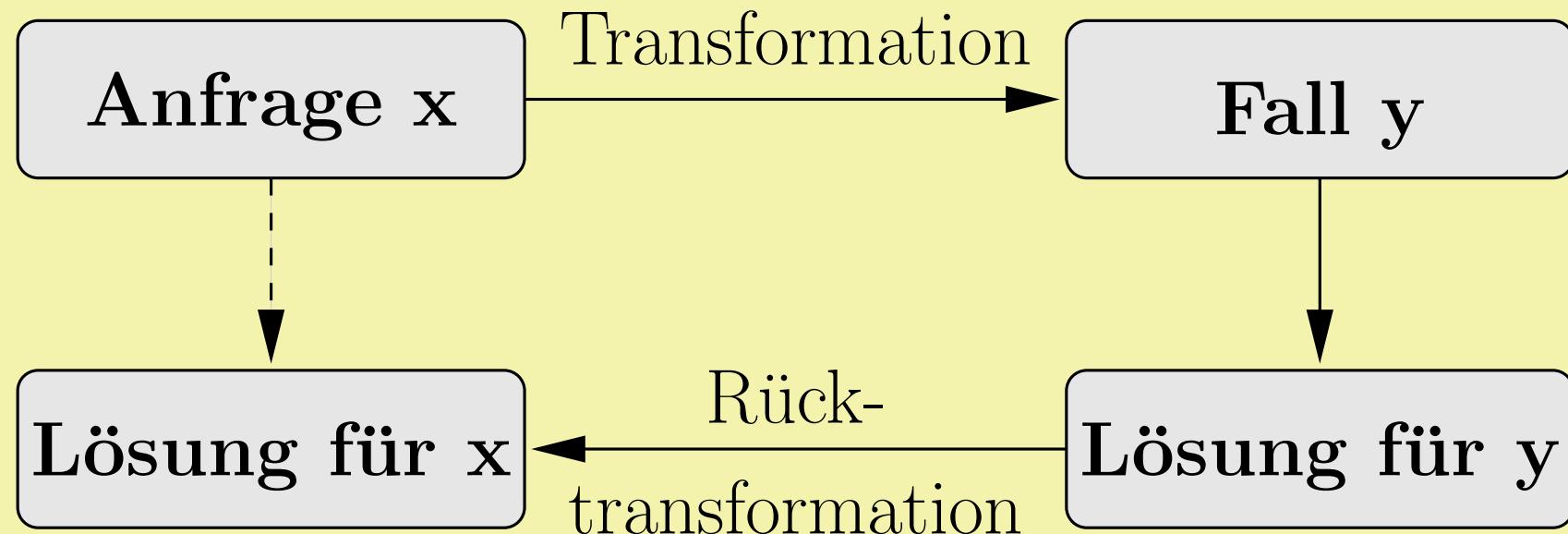
⁸Richter, M. Fallbasiertes Schließen. In Görz/Rollinger/Schneeberger Handbuch der Künstlichen Intelligenz.

CBR-Beispiel

Merkmal	Anfrage	Fall aus Fallbasis
Defektes Teil:	Rücklicht	Vorderlicht
Fahrrad Modell:	Marin Pine Mountain VSF T400	
Baujahr:	1993	2001
Stromquelle:	Batterie	Dynamo
Zustand der Birnen:	ok	ok
Lichtkabelzustand:	?	ok
Lösung		
Diagnose:	?	Massekontakt vorne fehlt
Reparatur:	?	Stelle Massekontakt vorne her

Transformation: Rücklicht auf Vorderlicht abbilden

CBR-Schema



CBR: Probleme

Modellierung Frame-Problem: Kann der Entwickler alle möglichen Spezialfälle und Problemvarianten vorhersehen und abbilden?

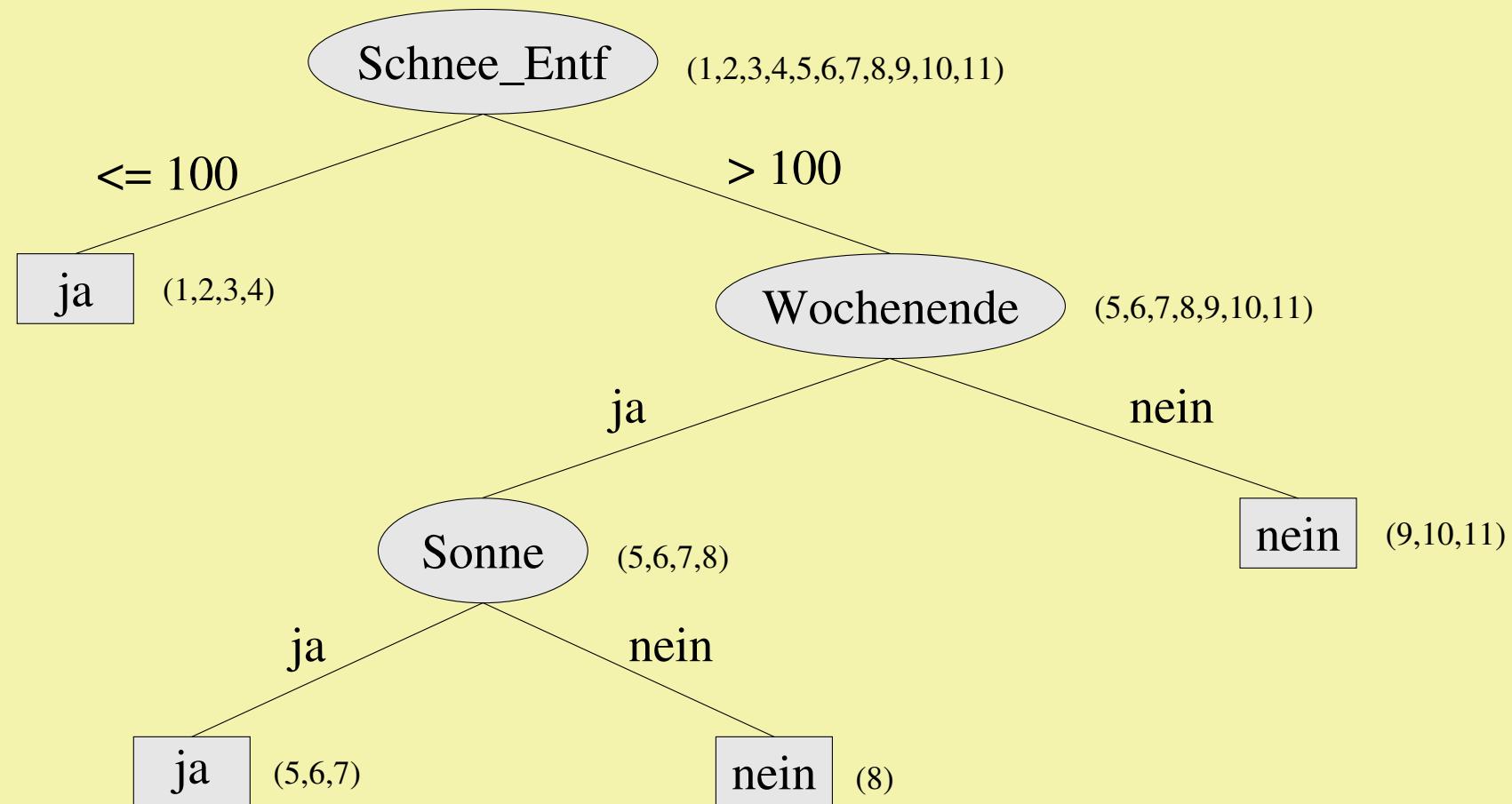
Ähnlichkeit passendes Ähnlichkeitsmass für symbolische Merkmale.

Transformation Wie findet man die Abbildung und deren Umkehrung?

Lösungen:

- Interessante Alternative zu CBR: Bayes-Netze.
- Symbolische Problemrepräsentation ist oft abbildbar auf numerische Merkmale.
- Dann sind andere Verfahren einsetzbar.

Lernen von Entscheidungsbäumen



Entscheidungsbaum für das Klassifikationsproblem Skifahren.

Bäume und Merkmale

Ein **Entscheidungsbaum** ist ein Baum, dessen innere Knoten Merkmale (Attribute) repräsentieren. Jede Kante steht für einen Attributwert. An jedem Blattknoten ist ein Klassenwert angegeben.

Variable	Werte	Beschreibung
Skifahren (Zielvariable)	ja, nein	Fahre ich los in das nächstgelegene Skigebiet mit ausreichend Schnee?
Sonne (Merkmal)	ja, nein	Sonne scheint heute?
Schnee_Entf (Merkmal)	≤ 100 , > 100	Entfernung des nächstes Skigebiets mit guten Schneeverhältnissen (über/unter 100 km)
Wochendende (Merkmal)	ja, nein	Ist heute Wochenende?

Variablen für das Klassifikationsproblem Skifahren.

Trainingsdaten

Tag Nr.	Schnee_Entf	Wochenende	Sonne	Skifahren
1	≤ 100	ja	ja	ja
2	≤ 100	ja	ja	ja
3	≤ 100	ja	nein	ja
4	≤ 100	nein	ja	ja
5	> 100	ja	ja	ja
6	> 100	ja	ja	ja
7	> 100	ja	ja	nein
8	> 100	ja	nein	nein
9	> 100	nein	ja	nein
10	> 100	nein	ja	nein
11	> 100	nein	nein	nein

Zeilen 6 und 7 widersprechen sich! Baum aus Abbildung 8.3 also optimal!

Wie entsteht der Baum aus den Daten?

optimaler Algorithmus:

alle Bäume erzeugen, für jeden Baum die Zahl der Fehlklassifikationen auf den Daten berechnen und schließlich einen Baum mit minimaler Fehlerzahl auswählen.

Nachteil: inakzeptabel hohe Rechenzeit!

also: heuristischer Algorithmus mit Greedy-Strategie

Die Entropie als Maß für den Informationsgehalt

Es soll das Attribut mit dem höchsten Informationsgewinn gewählt werden.

Trainingsdatenmenge $S = (\text{ja}, \text{ja}, \text{ja}, \text{ja}, \text{ja}, \text{ja}, \text{nein}, \text{nein}, \text{nein}, \text{nein}, \text{nein})$

mit geschätzten Wahrscheinlichkeiten

$$p_1 = p(\text{ja}) = 6/11 \quad \text{und} \quad p_2 = p(\text{nein}) = 5/11.$$

Wahrscheinlichkeitsverteilung

$$\mathbf{p} = (p_1, p_2) = (6/11, 5/11),$$

allgemein

$$\mathbf{p} = (p_1, \dots, p_n)$$

mit

$$\sum_{i=1}^n p_i = 1.$$

zwei Extremfälle

sicheres Ereignis:

$$\mathbf{p} = (1, 0, 0, \dots, 0). \quad (8.4)$$

Gleichverteilung:

$$\mathbf{p} = \left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n} \right) \quad (8.5)$$

Ungewissheit ist maximal!

Claude Shannon: wieviele Bits benötigt man mindestens, um solch ein Ereignis zu kodieren?

1. Fall (Gleichung 8.4): Null Bits
2. Fall (Gleichung 8.5): $\log_2 n$ Bits

allgemeiner Fall:

$$\mathbf{p} = (p_1, \dots, p_n) = \left(\frac{1}{m_1}, \frac{1}{m_2}, \dots, \frac{1}{m_n} \right) \quad (8.6)$$

$\log_2 m_i$ Bits für den i -ten Fall benötigt.

Erwartungswert H für die Zahl der Bits:

$$H = \sum_{i=1}^n p_i \log_2 m_i = \sum_{i=1}^n p_i \log_2 1/p_i = - \sum_{i=1}^n p_i \log_2 p_i.$$

Je mehr Bits man benötigt, um ein Ereignis zu kodieren, desto höher ist die Unsicherheit über den Ausgang.

Daher: **Entropie H** als Maß für die Unsicherheit einer Wahrscheinlichkeitsverteilung:

$$H(\mathbf{p}) = H(p_1, \dots, p_n) := - \sum_{i=1}^n p_i \log_2 p_i.$$

Problem: $0 \log_2 0$ undefiniert!

Definition $0 \log_2 0 := 0$ (siehe Aufgabe ??).

damit:

$$H(1, 0, \dots, 0) = 0$$

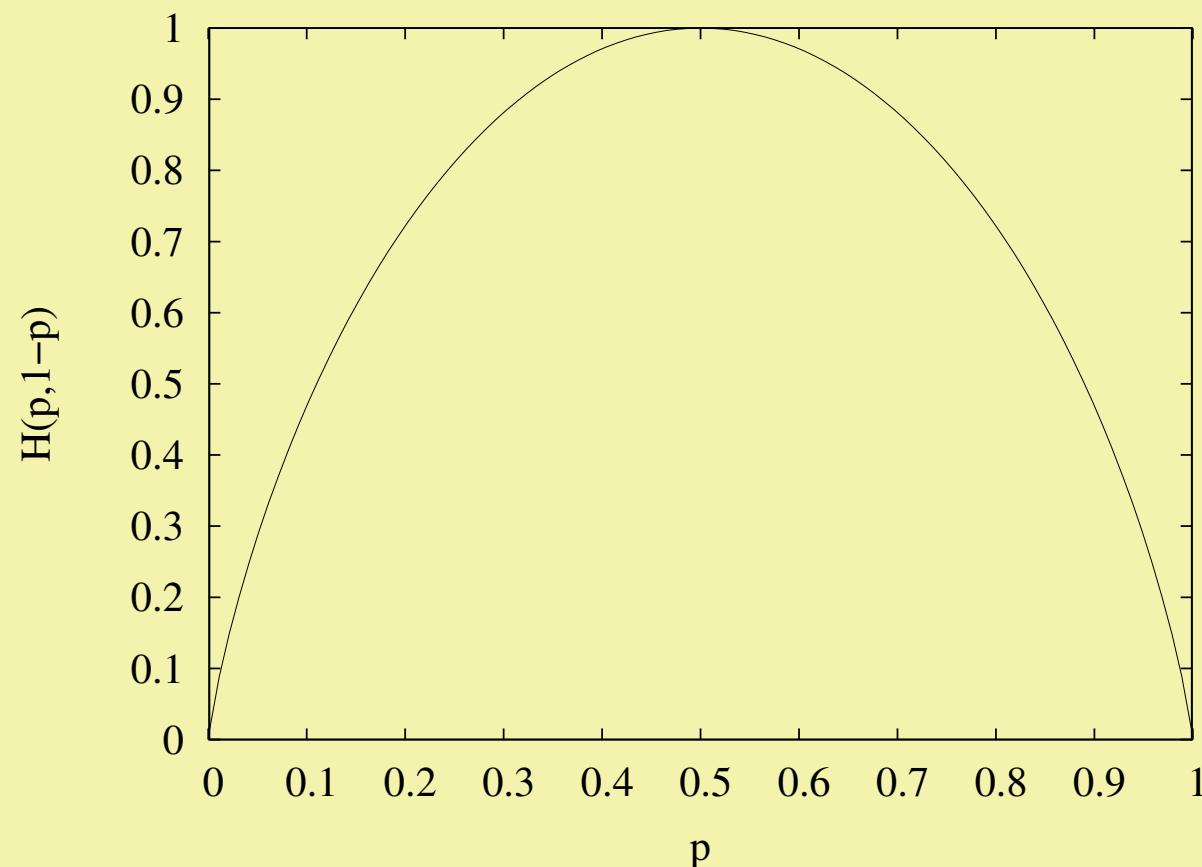
und

$$H\left(\frac{1}{n}, \dots, \frac{1}{n}\right) = \log_2 n$$

eindeutiges globales Maximum der Entropie!

Spezialfall: zwei Klassen

$$H(p) = H(p_1, p_2) = H(p_1, 1 - p_1) = -(p_1 \log_2 p_1 + (1 - p_1) \log_2(1 - p_1))$$



Die Entropiefunktion für den Zweiklassenfall.

Datenmenge D mit Wahrscheinlichkeitsverteilung p :

$$H(D) = H(p).$$

Informationsgehalt $I(D)$ der Datenmenge D ist das Gegenteil von Unsicherheit, also

$$I(D) := 1 - H(D). \quad (8.7)$$

Der Informationsgewinn

Wenden wir nun die Entropieformel an auf das Beispiel, so ergibt sich

$$H(6/11, 5/11) = 0.994$$

Informationsgewinn

$$\text{InfGewinn}(D, A) = \sum_{i=1}^n \frac{|D_i|}{|D|} I(D_i) - I(D)$$

Mit Gleichung 8.7 erhalten wir

$$\begin{aligned}
 \text{InfGewinn}(D, A) &= \sum_{i=1}^n \frac{|D_i|}{|D|} I(D_i) - I(D) = \sum_{i=1}^n \frac{|D_i|}{|D|} (1 - H(D_i)) - (1 - H(D)) \\
 &= 1 - \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) - 1 + H(D) \\
 &= H(D) - \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i)
 \end{aligned}$$

Angewendet auf Schnee_Entf:

$$\begin{aligned}
 \text{InfGewinn}(D, \text{Schnee_Entf}) &= H(D) - \left(\frac{4}{11} H(D_{\leq 100}) + \frac{7}{11} H(D_{> 100}) \right) \\
 &= 0.994 - \left(\frac{4}{11} \cdot 0 + \frac{7}{11} \cdot 0.863 \right) = 0.445
 \end{aligned}$$

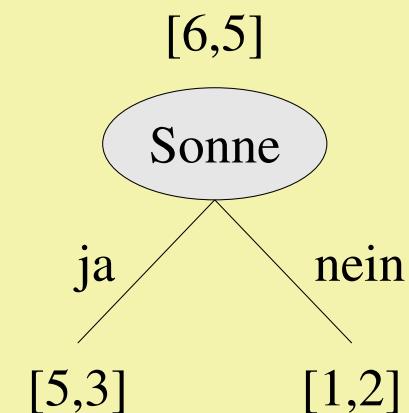
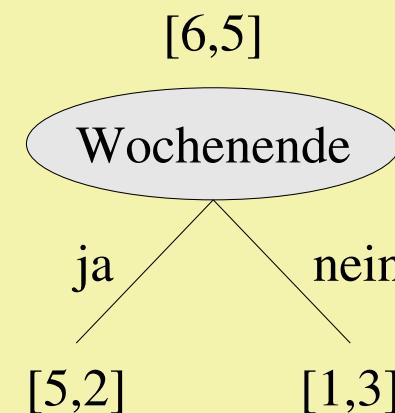
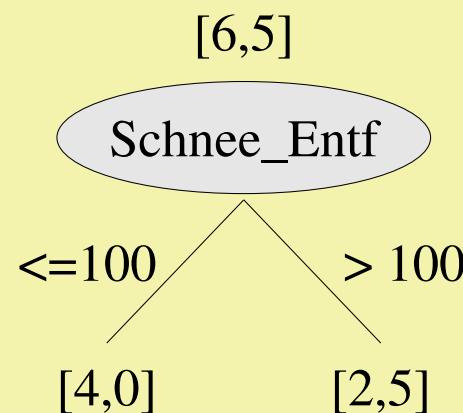
analog erhält man

$$\text{InfGewinn}(D, \text{Wochenende}) = 0.150$$

und

$$\text{InfGewinn}(D, \text{Sonne}) = 0.049$$

also: Wurzelknoten wird Schnee_Entf



Gewinn: 0.445

0.150

0.049

Vergleich

Tag Nr.	Schnee_Entf	Wochenende	Sonne	Skifahren
1	≤ 100	ja	ja	ja
2	≤ 100	ja	ja	ja
3	≤ 100	ja	nein	ja
4	≤ 100	nein	ja	ja
5	> 100	ja	ja	ja
6	> 100	ja	ja	ja
7	> 100	ja	ja	nein
8	> 100	ja	nein	nein
9	> 100	nein	ja	nein
10	> 100	nein	ja	nein
11	> 100	nein	nein	nein

Datenmenge für das Klassifikationsproblem Skifahren.

Für $D_{\leq 100}$ ist die Klassifikation eindeutig **ja**.

Also terminiert der Baum hier.

Im Zweig $D_{>100}$ herrscht keine Eindeutigkeit. Also

$$\text{InfGewinn}(D_{>100}, \text{Wochenende}) = 0.292$$

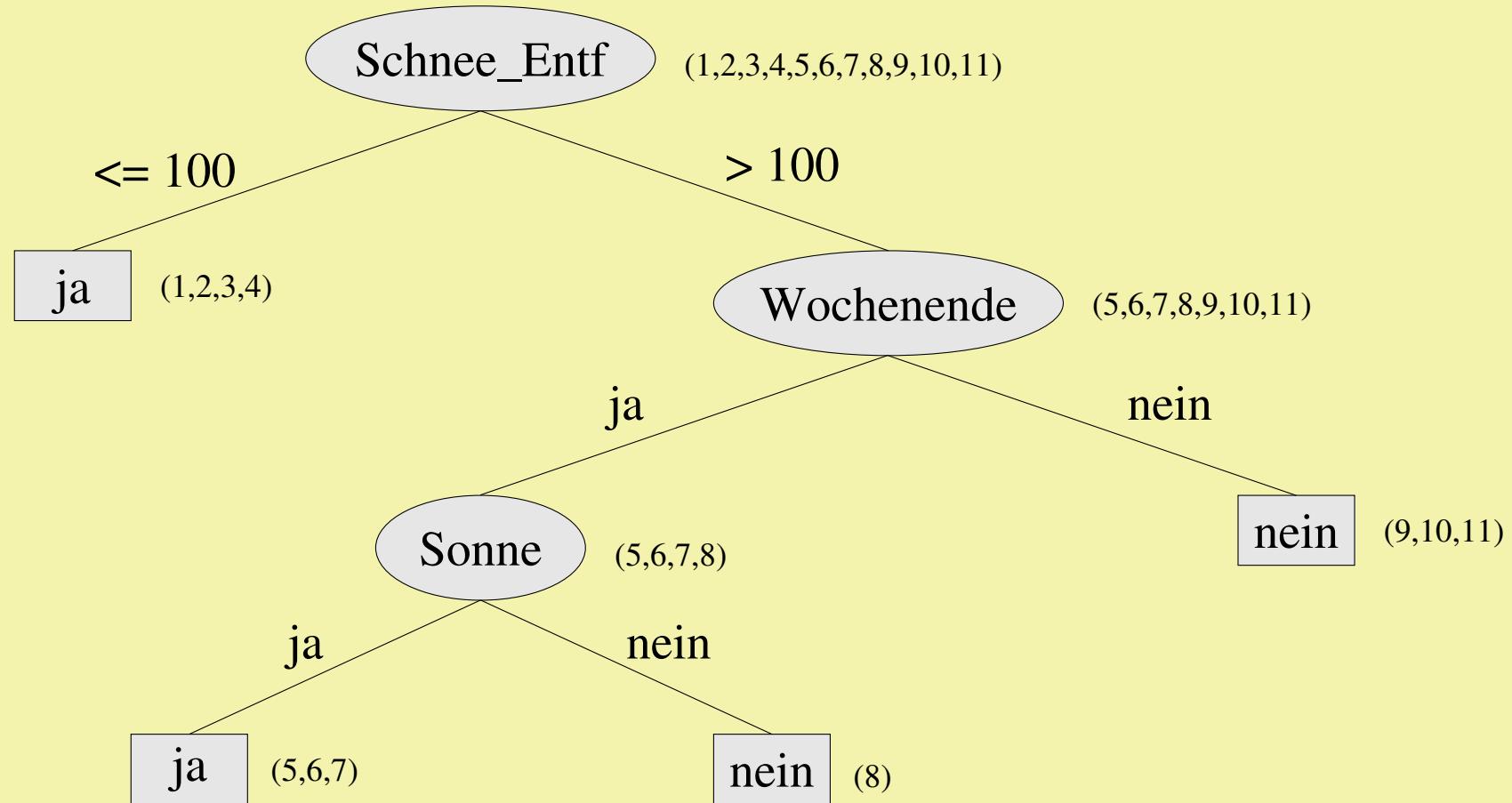
$$\text{InfGewinn}(D_{>100}, \text{Sonne}) = 0.170$$

Der Knoten erhält also das Attribut **Wochenende**.

Für **Wochenende = nein** terminiert der Baum mit der Entscheidung **nein**.

Für **Wochenende = ja** bringt **Sonne** einen Gewinn von 0.171.

Aufbau des Baumes beendet, weil keine weiteren Attribute mehr verfügbar sind.



Entscheidungsbaum für das Klassifikationsproblem Skifahren.

Systeme

C4.5 1993, Ross Quinlan, Weiterentwicklung von **ID3** (Iterative Dichotomiser 3, 1986). Nachfolger (kostenpflichtig): C5.0

CART (Classification and Regression Trees), 1984, Leo Breiman, ähnlich zu C4.5, komfortable graphische Benutzeroberfläche, teuer.

CHAID (Chi-square Automatic Interaction Detectors), 1964, J. Sonquist und J. Morgan, stoppt das Wachsen des Baumes bevor er zu groß wird. Heute bedeutungslos

Knime Data Mining Werkzeug (Konstanz Information Miner), sehr komfortable Benutzeroberfläche, Verwendet die **Weka** Java-Bibliothek.

Anwendung von C4.5

Trainingsdaten in Datei ski.data:

```
<=100, ja , ja , ja  
<=100, ja , ja , ja  
<=100, ja , nein, ja  
<=100, nein, ja , ja  
>100, ja , ja , ja  
>100, ja , ja , ja  
>100, ja , ja , nein  
>100, ja , nein, nein  
>100, nein, ja , nein  
>100, nein, ja , nein  
>100, nein, nein, nein
```

Informationen über Attribute und Klassen in Datei ski.names:

| Klassen: nein: nicht Skifahren, ja: Skifahren

|

nein,ja.

|

| Attribute

|

Schnee_Entf: <=100,>100.

Wochenende: nein,ja.

Sonne: nein,ja.

unixprompt> c4.5 -f ski -m 1

C4.5 [release 8] decision tree generator

Wed Aug 23 10:44:

Options:

File stem <ski>

Sensible test requires 2 branches with ≥ 1 cases

Read 11 cases (3 attributes) from ski.data

Decision Tree:

Schnee_Entf = ≤ 100 : ja (4.0)

Schnee_Entf = > 100 :

| Wochenende = nein: nein (3.0)

| Wochenende = ja:

| | Sonne = nein: nein (1.0)

| | Sonne = ja: ja (3.0/1.0)

Simplified Decision Tree:

Schnee_Entf = <=100: ja (4.0/1.2)

Schnee_Entf = >100: nein (7.0/3.4)

Evaluation on training data (11 items):

Before Pruning		After Pruning		
Size	Errors	Size	Errors	Estimate
7	1 (9.1%)	3	2 (18.2%)	(41.7%) <<

Der Lernalgorithmus

ERZEUGEENTSCHEIDUNGSBAUM($Daten, Knoten$)

A_{max} = Attribut mit maximalem Informationsgewinn

If InfGewinn(A_{max}) = 0

Then Knoten wird Blattknoten mit häufigster Klasse in Daten

Else ordne Knoten das Attribut A_{max} zu

Erzeuge für jeden Wert a_1, \dots, a_n von A_{max}

einen Nachfolgeknoten: K_1, \dots, K_n

Teile Daten auf in D_1, \dots, D_n mit $D_i = \{x \in Daten \mid A_{max}(x) = a_i\}$

For all $i \in \{1, \dots, n\}$

If alle $x \in D_i$ gehören zur gleichen Klasse C_i

Then Erzeuge Blattknoten K_i mit Klasse C_i

Else ERZEUGEENTSCHEIDUNGSBAUM(D_i, K_i)

Lernen von Appendizitisdiagnose (LEXMED)

app.names:

```
|Definition der Klassen und Attribute
|
|Klassen      0=Appendizitis negativ
|                  1=Appendizitis positiv
0,1.
|
|Attribute
|
Alter:                           continuous.
Geschlecht_(1=m__2=w):           1,2.
Schmerz_Quadrant1_(0=nein__1=ja): 0,1.
Schmerz_Quadrant2_(0=nein__1=ja): 0,1.
Schmerz_Quadrant3_(0=nein__1=ja): 0,1.
Schmerz_Quadrant4_(0=nein__1=ja): 0,1.
Lokale_Abwehrspannung_(0=nein__1=ja): 0,1.
Generalisierte_Abwehrspannung_(0=nein__1=ja): 0,1.
Schmerz_bei_Loslassmanoever_(0=nein__1=ja): 0,1.
Erschuetterung_(0=nein__1=ja): 0,1.
Schmerz_bei_rektaler_Untersuchung_(0=nein__1=ja): 0,1.
Temp_ax:                          continuous.
Temp_re:                          continuous.
```

Leukozyten: continuous.
Diabetes_mellitus_(0=nein__1=ja): 0,1

app.data:

```
19,1,0,0,1,0,1,0,1,1,0,362,378,13400,0,1  
13,1,0,0,1,0,1,0,1,1,1,383,385,18100,0,1  
32,2,0,0,1,0,1,0,1,1,0,364,374,11800,0,1  
18,2,0,0,1,1,0,0,0,0,0,362,370,09300,0,0  
73,2,1,0,1,1,1,0,1,1,1,376,380,13600,1,1  
30,1,1,1,1,1,0,1,1,1,1,377,387,21100,0,1  
56,1,1,1,1,1,0,1,1,1,0,390,?,14100,0,1  
36,1,0,0,1,0,1,0,1,1,0,372,382,11300,0,1  
36,2,0,0,1,0,0,0,1,1,1,370,379,15300,0,1  
33,1,0,0,1,0,1,0,1,1,0,367,376,17400,0,1  
19,1,0,0,1,0,0,0,1,1,0,361,375,17600,0,1  
12,1,0,0,1,0,1,0,1,1,0,364,370,12900,0,0  
...
```

Die Datei enthält 9764 Fälle.

```
unixprompt> c4.5 -f app -u -m 100
```

```
C4.5 [release 8] decision tree generator
```

```
-----  
Wed Aug 23 13:13:15 2006
```

Options:

```
File stem <app>  
Trees evaluated on unseen cases  
Sensible test requires 2 branches with >=100 cases
```

```
Read 9764 cases (15 attributes) from app.data
```

Decision Tree:

```
Leukozyten <= 11030 :
```

```
| Schmerz_bei_Loslassmanoever = 0:  
| | Temp_re > 381 : 1 (135.9/54.2)  
| | Temp_re <= 381 :  
| | | Lokale_Abwehrspannung = 0: 0 (1453.3/358.9)  
| | | Lokale_Abwehrspannung = 1:  
| | | | Geschlecht_(1=m____2=w) = 1: 1 (160.1/74.9)  
| | | | Geschlecht_(1=m____2=w) = 2: 0 (286.3/97.6)  
| Schmerz_bei_Loslassmanoever = 1:  
| | Leukozyten <= 8600 :
```

```
| | | Temp_re > 378 : 1 (176.0/59.4)
| | | Temp_re <= 378 :
| | |   | Geschlecht_(1=m____2=w) = 1:
| | |     | Lokale_Abwehrspannung = 0: 0 (110.7/51.7)
| | |     | Lokale_Abwehrspannung = 1: 1 (160.6/68.5)
| | |   | Geschlecht_(1=m____2=w) = 2:
| | |     | Alter <= 14 : 1 (131.1/63.1)
| | |     | Alter > 14 : 0 (398.3/137.6)
| | Leukozyten > 8600 :
| |   | Geschlecht_(1=m____2=w) = 1: 1 (429.9/91.0)
| |   | Geschlecht_(1=m____2=w) = 2:
| |   |   | Lokale_Abwehrspannung = 1: 1 (311.2/103.0)
| |   |   | Lokale_Abwehrspannung = 0:
| |   |     | Temp_re <= 375 : 1 (125.4/55.8)
| |   |     | Temp_re > 375 : 0 (118.3/56.1)
Leukozyten > 11030 :
| Schmerz_bei_Loslassmanoever = 1: 1 (4300.0/519.9)
| Schmerz_bei_Loslassmanoever = 0:
|   | Leukozyten > 14040 : 1 (826.6/163.8)
|   | Leukozyten <= 14040 :
|   |   | Erschuetterung = 1: 1 (260.6/83.7)
|   |   | Erschuetterung = 0:
|   |     | Lokale_Abwehrspannung = 1: 1 (117.5/44.4)
|   |     | Lokale_Abwehrspannung = 0:
```

| | | | | Temp_ax <= 368 : 0 (131.9/57.4)

| | | | | Temp_ax > 368 : 1 (130.5/57.8)

Simplified Decision Tree:

```
Leukozyten > 11030 : 1 (5767.0/964.1)
Leukozyten <= 11030 :
|   Schmerz_bei_Loslassmanoever = 0:
|   |   Temp_re > 381 : 1 (135.9/58.7)
|   |   Temp_re <= 381 :
|   |   |   Lokale_Abwehrspannung = 0: 0 (1453.3/370.9)
|   |   |   Lokale_Abwehrspannung = 1:
|   |   |   |   Geschlecht_(1=m____2=w) = 1: 1 (160.1/79.7)
|   |   |   |   Geschlecht_(1=m____2=w) = 2: 0 (286.3/103.7)
|   Schmerz_bei_Loslassmanoever = 1:
|   |   Leukozyten > 8600 : 1 (984.7/322.6)
|   |   Leukozyten <= 8600 :
|   |   |   Temp_re > 378 : 1 (176.0/64.3)
|   |   |   Temp_re <= 378 :
|   |   |   |   Geschlecht_(1=m____2=w) = 1:
|   |   |   |   |   Lokale_Abwehrspannung = 0: 0 (110.7/55.8)
|   |   |   |   |   Lokale_Abwehrspannung = 1: 1 (160.6/73.4)
|   |   |   |   Geschlecht_(1=m____2=w) = 2:
|   |   |   |   |   Alter <= 14 : 1 (131.1/67.6)
|   |   |   |   |   Alter > 14 : 0 (398.3/144.7)
```


Evaluation on training data (9764 items):

Before Pruning		After Pruning		
Size	Errors	Size	Errors	Estimate
37	2197(22.5%)	21	2223(22.8%)	(23.6%) <<

Evaluation on test data (4882 items):

Before Pruning		After Pruning		
Size	Errors	Size	Errors	Estimate
37	1148(23.5%)	21	1153(23.6%)	(23.6%) <<
(a) (b)		<-classified as		
758	885		(a): class 0	
268	2971		(b): class 1	

Stetige Attribute

Knoten: Leukozyten > 11030

Die Schwelle $\Theta_{D,A}$ für ein Attribut A wird bestimmt durch

$$\Theta_{D,A} = \max_v \{\text{InfGewinn}(D, A > v)\}.$$

für alle in den Trainingsdaten D vorkommenden Werte v von A .

Jedes stetige Attribut wird an jedem neu erzeugten Knoten wieder getestet!

Pruning – Der Baumschnitt

Aristoteles:

von zwei wissenschaftlichen Theorien, die den gleichen Sachverhalt gleich gut erklären, ist die einfachere zu bevorzugen.

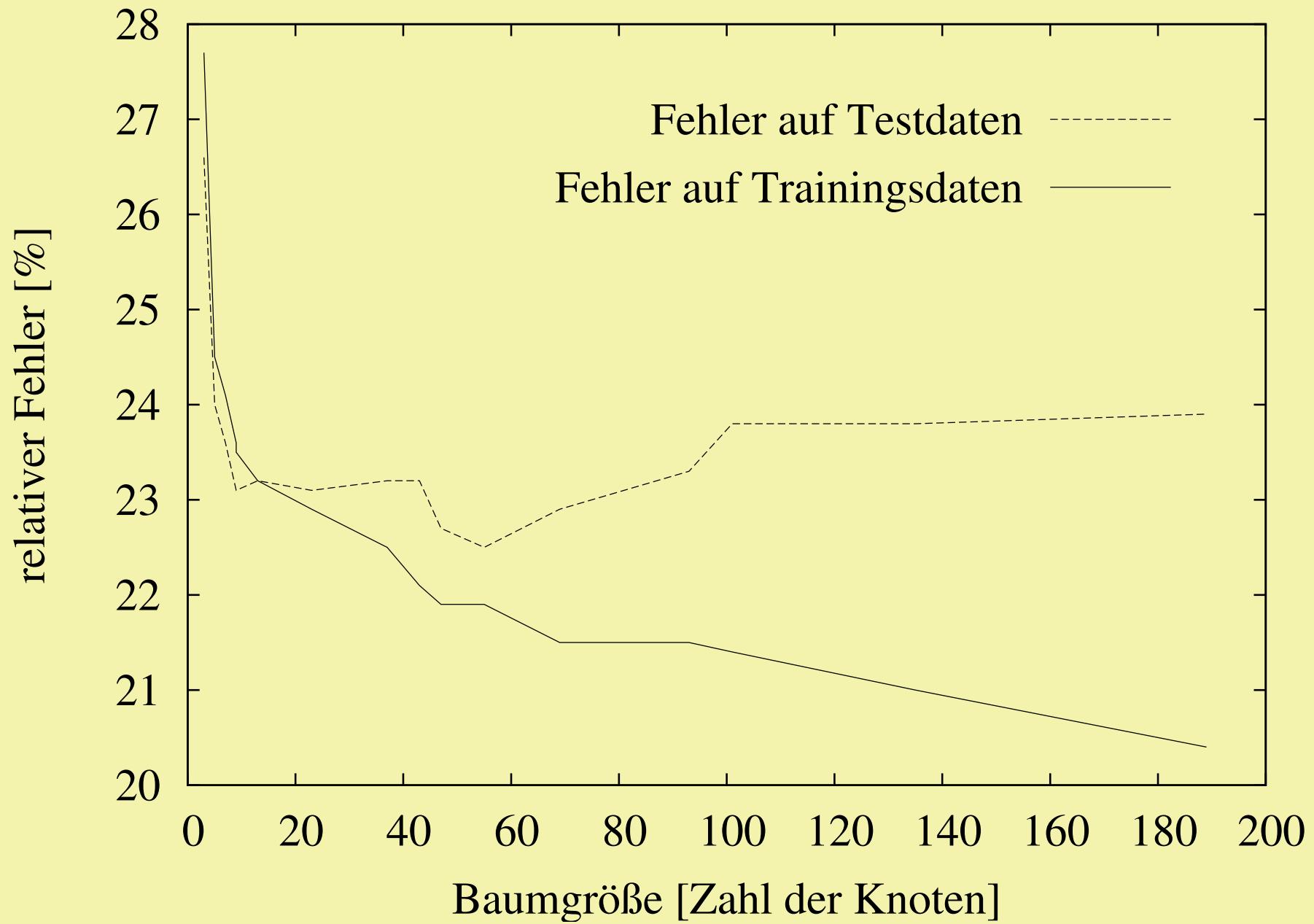
Occam's Rasiermesser (engl. Occam's razor)

Unter allen Bäumen mit einer festen Fehlerrate ist also immer ein kleinster Baum auszuwählen.

Generalisieren

- Je komplexer ein Modell, desto mehr Details werden repräsentiert, aber umso schlechter lässt sich das Modell auf neue Daten übertragen. Daher:
- Trainingsdaten (z.B. 2/3 aller Daten)
- Testdaten (z.B. 1/3 aller Daten)

Überanpassung (engl. overfitting)



Überanpassung

Definition 8.10 Sei ein bestimmtes Lernverfahren, das heißt eine Klasse lernender Agenten, gegeben. Man nennt einen Agenten A überangepasst an die Trainingsdaten, wenn es einen anderen Agenten A' gibt, dessen Fehler auf den Trainingsdaten größer ist als der von A , aber auf der gesamten Verteilung von Daten ist der Fehler von A' kleiner als der von A .

Kreuzvalidierung (engl. cross validation):

Fehler auf den Testdaten messen bis er signifikant ansteigt!

Pruning bei C4.5

Reduced Error Pruning schneidet solange Knoten des Baumes ab, bis ein aus dem Fehler auf den Trainingsdaten geschätzter Fehler auf den Testdaten wieder zunimmt.⁹

Greedy-Verfahren:

- nicht optimal,
- Bias hin zu kleinen Bäumen mit Attributen mit hohem Gain ganz oben.
- Gain ratio bei vielen Attributwerten.
- Stärken: sehr schnell, bei den Lexmed-Daten (ca. 10000 Trainingsdaten mit 15 Attributen) ca. 0.18 sec

⁹Besser wäre es allerdings, beim Pruning den Fehler auf den Testdaten zu verwenden.

Fehlende Werte

56, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 390, ?, 14100, 0, 1,

Möglichkeiten beim Aufbau des Baumes: verwende

- häufigsten Wert unter allen anderen Trainingsdaten
- häufigsten Wert unter allen anderen Trainingsdaten in der gleichen Klasse
- Wahrscheinlichkeitsverteilung aller Attributwerte einsetzen und das fehlerhafte Trainingsbeispiel entsprechend dieser Verteilung aufteilen auf die verschiedenen Äste im Baum¹⁰

Bei der Klassifikation werden fehlende Werte gleich behandelt wie beim Lernen.

¹⁰Grund für das Vorkommen von nicht ganzzahligen Werten in den Klammerausdrücken hinter den Blattknoten der C4.5-Bäume.

Zusammenfassung

- Für Klassifikationsaufgaben sehr beliebtes Verfahren
- Einfache Anwendung
- hohe Geschwindigkeit: Auf 10 000 Daten mit 15 Attributen benötigt C4.5 etwa 0.3 Sekunden für das Lernen
- Irrelevante Attribute werden automatisch gelöscht
- Anwender kann den Entscheidungsbaum verstehen
- Anwender kann den Entscheidungsbaum verändern
- Entscheidungsbaum kann leicht in bestehende Programme eingebaut werden
- erzeugte Bäume sind nicht immer optimal¹¹

¹¹Mitchell, T. Machine Learning. McGraw Hill, 1997.

Lernen von Bayes-Netzen

Lernen der Netzwerkstruktur

Bei vorgegebenen Variablen wird aus den Trainingsdaten die Netzwerktopologie generiert. Dieser erste Schritt ist der mit Abstand schwierigere und verdient im Folgenden eine genauere Betrachtung.

Lernen der bedingten Wahrscheinlichkeiten

Bei bekannter Netzwerktopologie müssen die CPTs mit Werten gefüllt werden. Wenn genügend Trainingsdaten verfügbar sind, können durch Zählen von Häufigkeiten in den Daten alle benötigten bedingten Wahrscheinlichkeiten geschätzt werden. Dieser Schritt ist relativ leicht automatisierbar.

Lernen der Netzwerkstruktur

Siehe Jensen¹²

- kausale Abhangigkeit kann nicht einfach automatisch erkannt werden
- Finden einer optimalen Struktur fur ein Bayes-Netz als klassisches Suchproblem:
 - Gegeben: Variablen V_1, \dots, V_n , Datei mit Trainingsdaten.
 - Gesucht: gerichteter azyklischer Graph (engl. directed acyclic graph, DAG), welcher die den Daten zugrunde liegende Wahrscheinlichkeitsverteilung mglichst gut wiedergibt.

Suchraum: Zahl unterschiedlicher DAGs wachst schnell:

Bei 5 Knoten 29281 DAGs

bei 9 Knoten 10^{15} DAGs¹³.

¹²Jensen, F.V. Bayesian networks and decision graphs. Springer-Verlag, 2001.

¹³Melancon, G./Dutour, I./Bousque-Melou, G. Random Generation of Dags for Graph Drawing. Dutch Research Center for Mathematical and Computer Science (CWI), 2000 (INS-R0005). – Technischer Bericht.

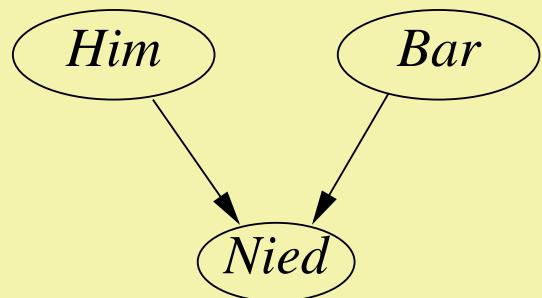
Heuristische Verfahren

- Suche im Raum aller DAGs
- schätze für jeden DAG die Werte der CPTs
- berechnen daraus die Verteilung
- Berechne Abstand der Verteilung zu der aus den Daten bekannten Verteilung
- Wähle die Verteilung mit dem kleinsten Abstand zu der aus den Daten bekannten Verteilung

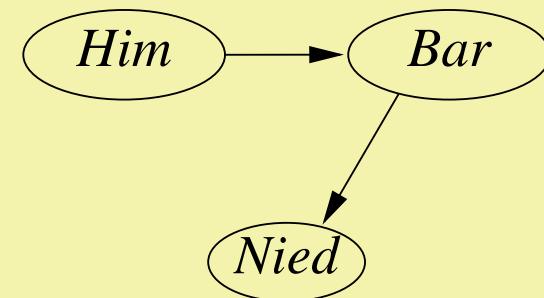
Beispiel: Wettervorhersage aus Aufgabe ??

$$\mathbf{P}(Him, Bar, Nied) = (0.40, 0.07, 0.08, 0.10, 0.09, 0.11, 0.03, 0.12).$$

Netz a:



Netz b:



euklidische Norm der Differenz der beiden Vektoren:

$$d_q(\mathbf{x}, \mathbf{y}) = \sum_i (x_i - y_i)^2$$

Abstand für Beispiel: $d_q(\mathbf{P}_a, \mathbf{P}) = 0.0029$, $d_q(\mathbf{P}_b, \mathbf{P}) = 0.014$.

Netz a besser!

Alternative (Kullback-Leibler-Abstand): $d_k(\mathbf{x}, \mathbf{y}) = \sum_i y_i (\log_2 y_i - \log_2 x_i)$,

Abstand für Beispiel: $d_k(\mathbf{P}_a, \mathbf{P}) = 0.017$, $d_k(\mathbf{P}_b, \mathbf{P}) = 0.09$

Gefahr: Überanpassung

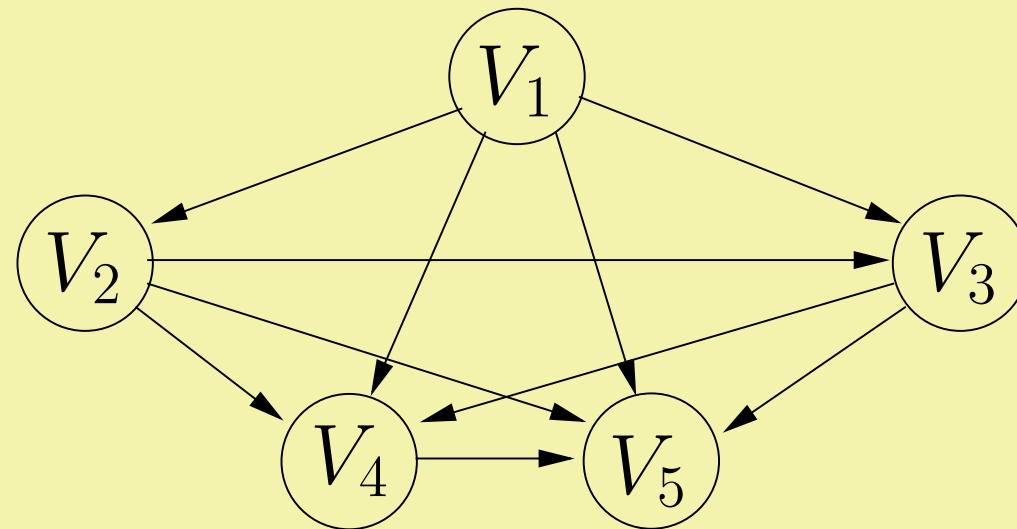
größeres Gewicht für kleine Netze:

$$f(N) = \text{Größe}(N) + w \cdot d_k(\mathbf{P}_N, \mathbf{P}).$$

Hierbei ist $\text{Größe}(N)$ die Anzahl aller Einträge in den CPTs und \mathbf{P}_N die Verteilung des Netzes N . w ist ein Gewichtungsfaktor, der manuell angepasst werden muss.

Suchraumproblem:

ausgehend von einer (kausalen) Ordnung der Variablen V_1, \dots, V_n nur diejenigen gerichteten Kanten (V_i, V_j) in das Netz einzeichnen, für die $i < j$. Gestartet wird mit dem maximalen Modell, das diese Bedingung erfüllt (5 Variablen):



gierige Suche: Kanten entfernen, bis f nicht mehr weiter abnimmt.

Forschung zum Lernen von Bayes-Netzen

- EM-Algorithmus
- Markov-Chain-Monte-Carlo-Methoden
- Gibbs-Sampling
- Hugin (www.hugin.com) oder
- Bayesware (www.bayesware.com).

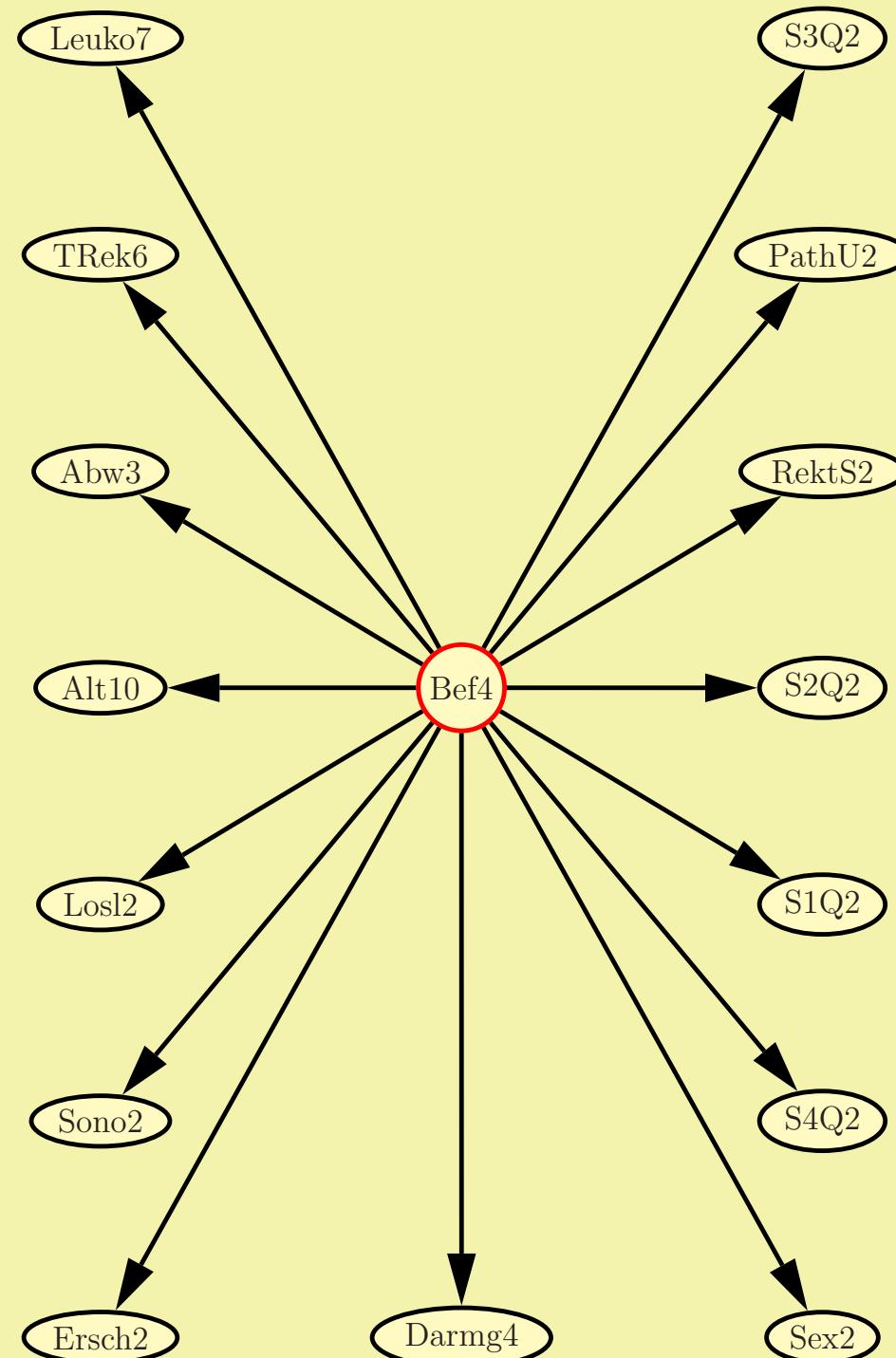
Der Naive-Bayes-Klassifizierer

$$P(B|S_1, \dots, S_n) = \frac{P(S_1, \dots, S_n|B) \cdot P(B)}{P(S_1, \dots, S_n)},$$

Worst-Case: 20 643 840 Wahrscheinlichkeiten $\mathbf{P}(S_1, \dots, S_n, B)$ müssen bestimmt werden!

Annahme: alle Symptomvariablen sind bedingt unabhängig gegeben B :

$$P(S_1, \dots, S_n|B) = P(S_1|B) \cdot \dots \cdot P(S_n|B).$$



Damit erhält man die Formel

$$P(B|S_1, \dots, S_n) = \frac{P(B) \prod_{i=1}^n P(S_i|B)}{P(S_1, \dots, S_n)}. \quad (8.8)$$

Naive-Bayes-Klassifizierer:

$$b_{\text{Naive-Bayes}} = \operatorname{argmax}_{i \in \{1, \dots, k\}} P(B = b_i | S_1, \dots, S_n).$$

Nenner in Gleichung 8.8 konstant, also:

Naive-Bayes-Formel

$$b_{\text{Naive-Bayes}} = \operatorname{argmax}_{i \in \{1, \dots, k\}} P(B = b_i) \prod_{i=1}^n P(S_i|B)$$

Anzahl der Werte in den CPTs:

$$6 \cdot 4 + 5 \cdot 4 + 2 \cdot 4 + 9 \cdot 4 + 3 \cdot 4 + 10 \cdot (1 \cdot 4) + 1 = 141.$$

- Für LEXMED wäre diese Vereinfachung nicht mehr akzeptabel!
- Naive-Bayes-Klassifikation ist gleich ausdrucksstark wie lineare Scores
- Scores liegt die Annahme zugrunde, dass alle Symptome bedingt unabhängig gegeben der Befund sind.

Schätzen von Wahrscheinlichkeiten

Problem, wenn die $P(S_i|B)$ klein sind!

$$P(S_i = x|B = y) = \frac{|S_i = x \wedge B = y|}{|B = y|}.$$

Beispiel: $P(S_i = x|B = y) = 0.01$, 40 Trainingsfälle $B = y$.

sehr wahrscheinlich wird $P(S_i = x|B = y) = 0$ geschätzt.

Aber $P(S_i = x|B = z) > 0$

Damit wird der Wert $B = z$ immer bevorzugt. **Fehler!**

Daher wird

$$P(A|B) \approx \frac{|A \wedge B|}{|B|} = \frac{n_{AB}}{n_B}$$

ersetzt durch

$$P(A|B) \approx \frac{n_{AB} + mp}{n_B + m},$$

wobei $p = P(A)$

Je größer m wird, desto größer das Gewicht der A-priori-Wahrscheinlichkeit im Vergleich zu dem aus der gemessenen Häufigkeit bestimmten Wert.

Textklassifikation mit Naive-Bayes

- Naive-Bayes wird oft verwendet zur Textklassifikation
- automatisches Filtern von Email (Spam-Filter)
- Naive-Bayes-Klassifizierer lernt, erwünschte Mails von Spam zu trennen.
- Ausfiltern von unerwünschten Beiträgen in Internet-Diskussionsforen
- Aufspüren von Webseiten mit kriminellen Inhalten
- Suche in Volltext-Datenbanken oder in der Literatur nach relevanten Informationen und Filterung

Lernen eines Spam-Filters:

- Der Benutzer muss eine größere Anzahl Emails manuell als erwünscht oder Spam klassifizieren.
- Der Filter wird trainiert.
- Wiederholung von 1 und 2

Beispiel: Textanalyse

Beispieltext von Alan Turing¹⁴:

„Wir dürfen hoffen, dass Maschinen vielleicht einmal auf allen rein intellektuellen Gebieten mit dem Menschen konkurrieren. Aber mit welchem sollte man am besten beginnen? Viele glauben, dass eine sehr abstrakte Tätigkeit, beispielsweise das Schachspielen, am besten geeignet wäre. Ebenso kann man behaupten, dass es das beste wäre, Maschinen mit den besten Sinnesorganen auszustatten, die überhaupt für Geld zu haben sind, und sie dann zu lehren, englisch zu verstehen und zu sprechen. Dieser Prozess könnte sich wie das normale Unterrichten eines Kindes vollziehen. Dinge würden erklärt und benannt werden, usw. Wiederum weiß ich nicht, welches die richtige Antwort ist, aber ich meine, dass man beide Ansätze erproben sollte.“

- Textklassen: „ I “ für interessant und „ $\neg I$ “ für uninteressant.
- Datenbank schon klassifizierter Texte.
- Welche Attribute sollen verwendet werden?

¹⁴Turing, A.M. Computing Machinery and Intelligence. Mind, 59 1950.

Naive Bayes macht das so:

Für jede der n Wortpositionen im Text wird ein Attribut s_i definiert. Als mögliche Werte für alle Positionen s_i sind alle im Text vorkommenden Worte erlaubt.

Nun müssen die Werte

$$P(I|s_1, \dots, s_n) = c \cdot P(I) \prod_{i=1}^n P(s_i|I) \quad (8.9)$$

sowie $P(\neg I|s_1, \dots, s_n)$ berechnet werden.

In obigem Beispiel mit insgesamt 107 Worten ergibt dies

$$\begin{aligned} P(I|s_1, \dots, s_n) &= \\ &c \cdot P(I) \cdot P(s_1 = \text{"Wir"}|I) \cdot P(s_2 = \text{"dürfen"}|I) \cdot \dots \cdot P(s_{107} = \text{"sollte"}|I) \end{aligned}$$

und

$$\begin{aligned} P(\neg I|s_1, \dots, s_n) &= \\ &c \cdot P(\neg I) \cdot P(s_1 = \text{"Wir"}|\neg I) \cdot P(s_2 = \text{"dürfen"}|\neg I) \cdot \dots \cdot P(s_{107} = \text{"sollte"}|\neg I). \end{aligned}$$

Wie lernt Naive Bayes?

auf den beiden Klassen von Texten werden alle

$$P(s_i|K)$$

und die A-priori-Wahrscheinlichkeiten

$$P(I), P(\neg I)$$

berechnet.

Die Naive Bayes-Annahme:

$P(s_i|I)$ hängt nicht von der Position im Text ab!!!

Das heißt zum Beispiel, dass

$$P(s_{61} = \text{"und"}|I) = P(s_{69} = \text{"und"}|I) = P(s_{86} = \text{"und"}|I).$$

einfacher: $P(\text{und}|I)$

Für jedes im Text vorkommende Wort w_i wird seine Häufigkeit n_i ermittelt und Gleichung 8.9 vereinfacht sich:

$$P(I|s_1, \dots, s_n) = c \cdot P(I) \prod_{i=1}^l P(w_i|I)^{n_i} \quad (8.10)$$

Man beachte, dass der Index i im Produkt nun nur noch bis zur Zahl l unterschiedlicher im Text vorkommender Worte läuft.

Ergebnisse

- Naive-Bayes liefert bei der Textklassifikation hervorragende Ergebnisse
- Spam-Filter erreichen Fehlerraten von weit unter einem Prozent.
- Die Systeme DSPAM und CRM114 erreichen eine Korrektheit von fast 99,99%¹⁵.

¹⁵Zdziarski, J. Ending Spam. No Starch Press, 2005.

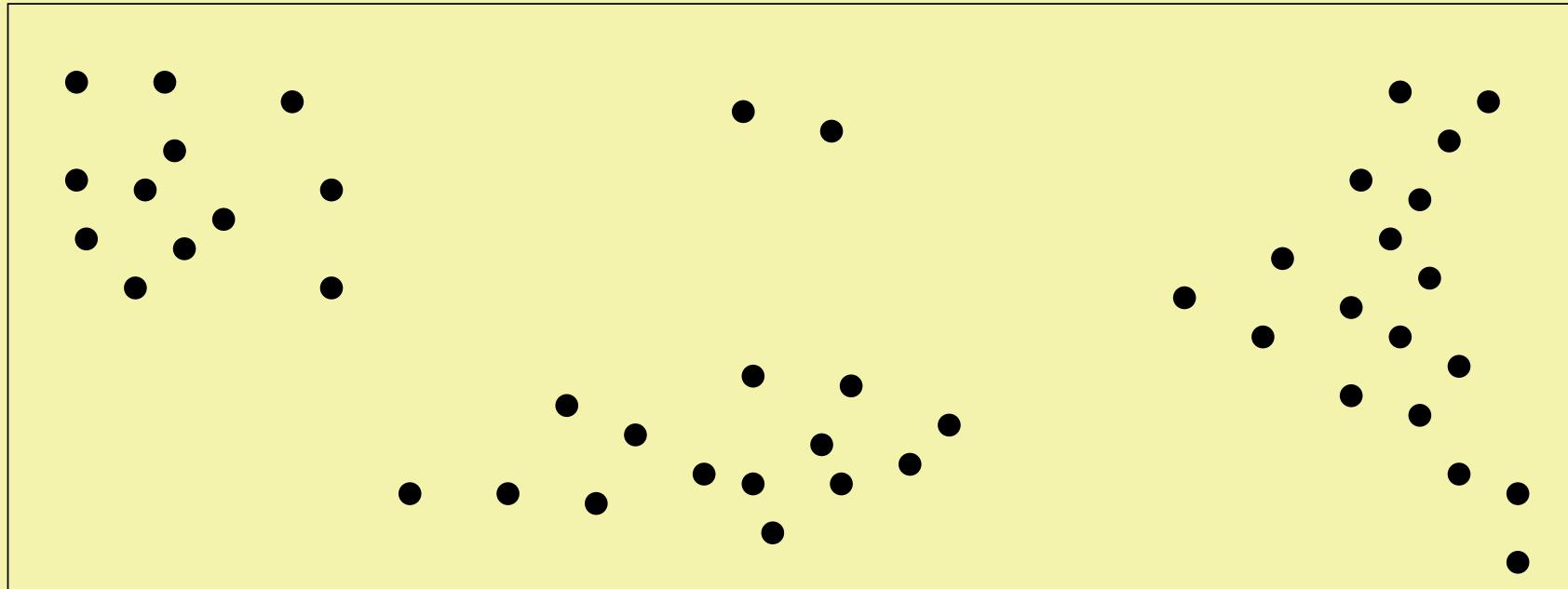
Clustering

Suchmaschineneingabe: „Decke“

Treffer:

- „Microfaser-Decke“,
- „unter der Decke“
- „Stuck an der Decke“
- „Wie streiche ich eine Decke“

zwei unterscheidbare **Cluster!**



Einfaches zweidimensionales Beispiel mit vier deutlich getrennten Clustern.

- **Clustering:** Trainingsdaten sind nicht klassifiziert.
- Der Lehrer fehlt hier
- Finden von Strukturen
- Häufungen im Raum der Trainingsdaten finden
- Ganz wichtig: Wahl eines geeigneten Abstandsmaßes für Punkte

Abstandsmaße

der Euklidische Abstand

$$d_e(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}.$$

Summe der Abstandsquadrate

$$d_q(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n (x_i - y_i)^2,$$

Manhattan-Abstand

$$d_m(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n |x_i - y_i|$$

Abstand der maximalen Komponente

$$d_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1, \dots, n} |x_i - y_i|$$

das normierte Skalarprodukt

$$\frac{x \cdot y}{|x| \cdot |y|}$$

als Abstandsmaß zum Beispiel der Kehrwert

$$d_s(x, y) = \frac{|x| \cdot |y|}{x \cdot y}$$

Beispiel: Suche in Volltextdatenbanken

Merkmale x_1, \dots, x_n als Komponenten des Vektors \boldsymbol{x} , werden wie folgt berechnet:

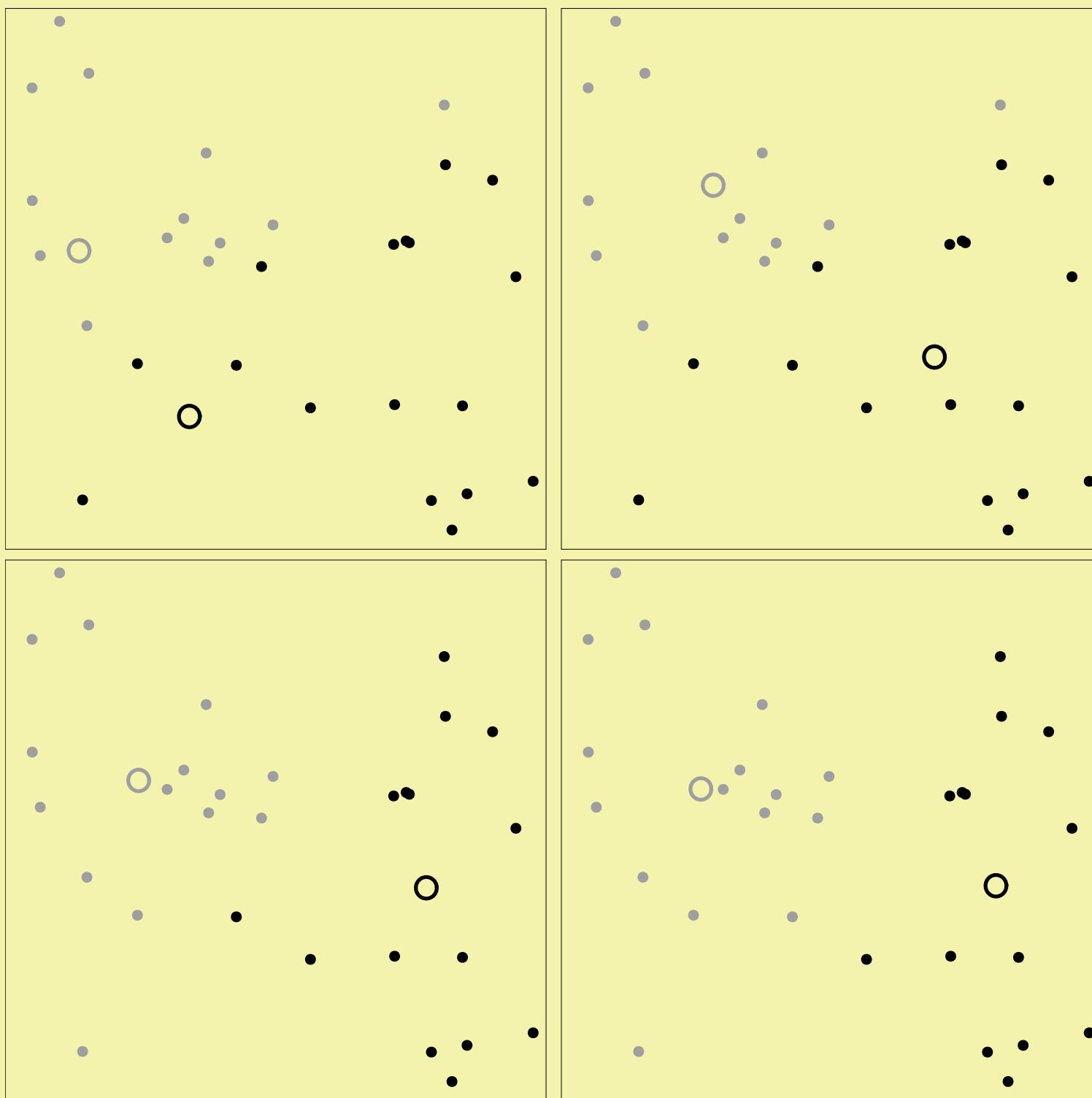
- Wörterbuch mit zum Beispiel 50000 Wörtern
- Vektor hat die Länge 50000
- $x_i = \text{Häufigkeit des } i\text{-ten Wörterbuch-Wortes im Text.}$

fast alle Komponenten sind null

k-Means und der EM-Algorithmus

Zahl der Cluster ist bekannt! **k-Means**-Verfahren:

- Zuerst werden die k Clustermittelpunkte μ_1, \dots, μ_k zufällig oder manuell initialisiert.
- Dann wird wiederholt:
 - Klassifikation aller Daten zum nächsten Clustermittelpunkt
 - Neuberechnung der Clustermittelpunkte



Algorithmus

K-MEANS(x_1, \dots, x_n, k)

initialisiere μ_1, \dots, μ_k (z.B. zufällig)

Repeat

Klassifiziere x_1, \dots, x_n zum jeweils nächsten μ_i

Berechne μ_1, \dots, μ_k neu

Until keine Änderung in μ_1, \dots, μ_k

Return(μ_1, \dots, μ_k)

Berechnung des Clustermittelpunktes:

$$\mu = \frac{1}{l} \sum_{i=1}^l x_i.$$

- keine Konvergenzgarantie, aber meist recht schnelle Konvergenz.
- Zahl der Iterationsschritte meist viel kleiner als Zahl der Punkte.
- Komplexität: $O(ndkt)$, mit
 - n = Gesamtzahl der Punkte,
 - d = Dimension des Merkmalsraumes
 - t = Zahl der Iterationsschritte

EM-Algorithmus

- stetige Variante von k-Means
- liefert für jeden Punkt die Wahrscheinlichkeiten für die Zugehörigkeit zu den verschiedenen Klassen.
- Art der Wahrscheinlichkeitsverteilung der Daten ist bekannt. (Oft Normalverteilung)
- EM-Algorithmus bestimmt die Parameter (Mittelwert μ und Standardabweichungen σ_{ij} der k mehrdimensionalen Normalverteilungen) für jeden Cluster:

Expectation: Für jeden Datenpunkt wird berechnet, mit welcher Wahrscheinlichkeit $P(C_j|x_i)$ er zu jedem der Cluster gehört.

Maximization: Unter Verwendung der neu berechneten Wahrscheinlichkeiten werden die Parameter der Verteilung neu berechnet.

weicheres Clustering!

Der EM-Algorithmus wird neben dem Clustering zum Beispiel für das Lernen von Bayes-Netzen eingesetzt¹⁶.

¹⁶Duda, R.O./Hart, P.E./Stork, D.G. Pattern Classification. Wiley, 2001.

Hierarchisches Clustering

Anfang: n Cluster mit je einem Punkt. Wiederholt werden die beiden nächsten Nachbarcluster vereinigt:

HIERARCHISCHES-CLUSTERING($\mathbf{x}_1, \dots, \mathbf{x}_n$)

initialisiere $C_1 = \{\mathbf{x}_1\}, \dots, C_n = \{\mathbf{x}_n\}$

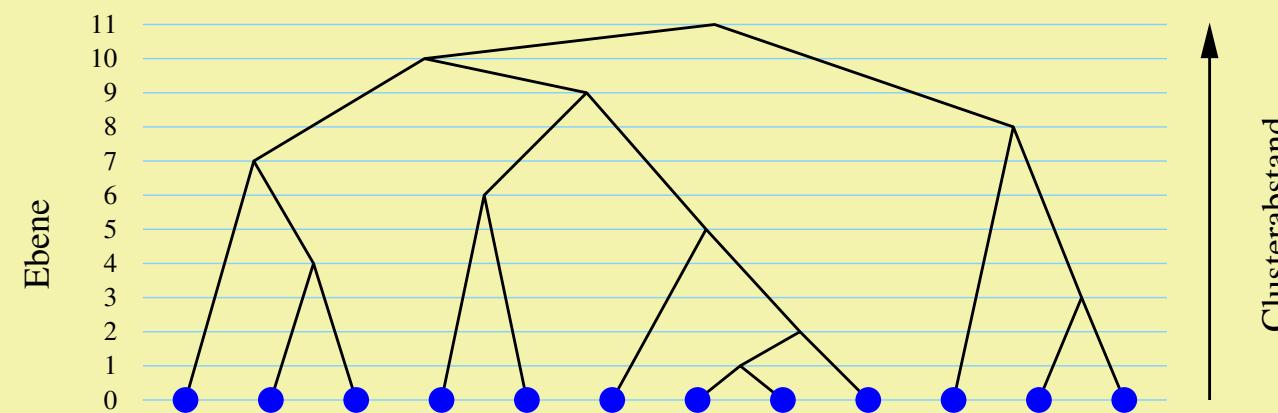
Repeat

 Finde zwei Cluster C_i und C_j mit kleinstem Abstand

 Vereinige C_i und C_j

Until Abbruchbedingung erreicht

Return(Baum mit Clustern)



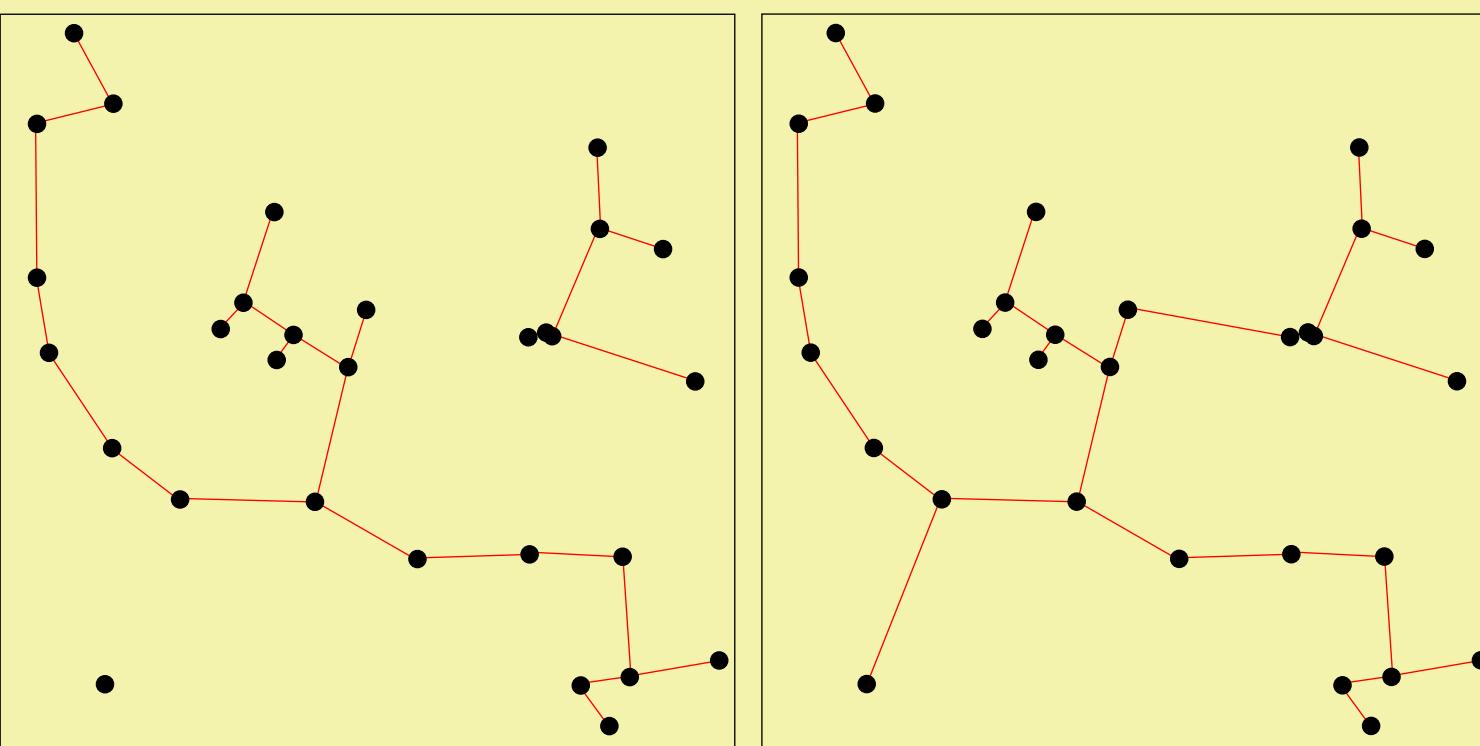
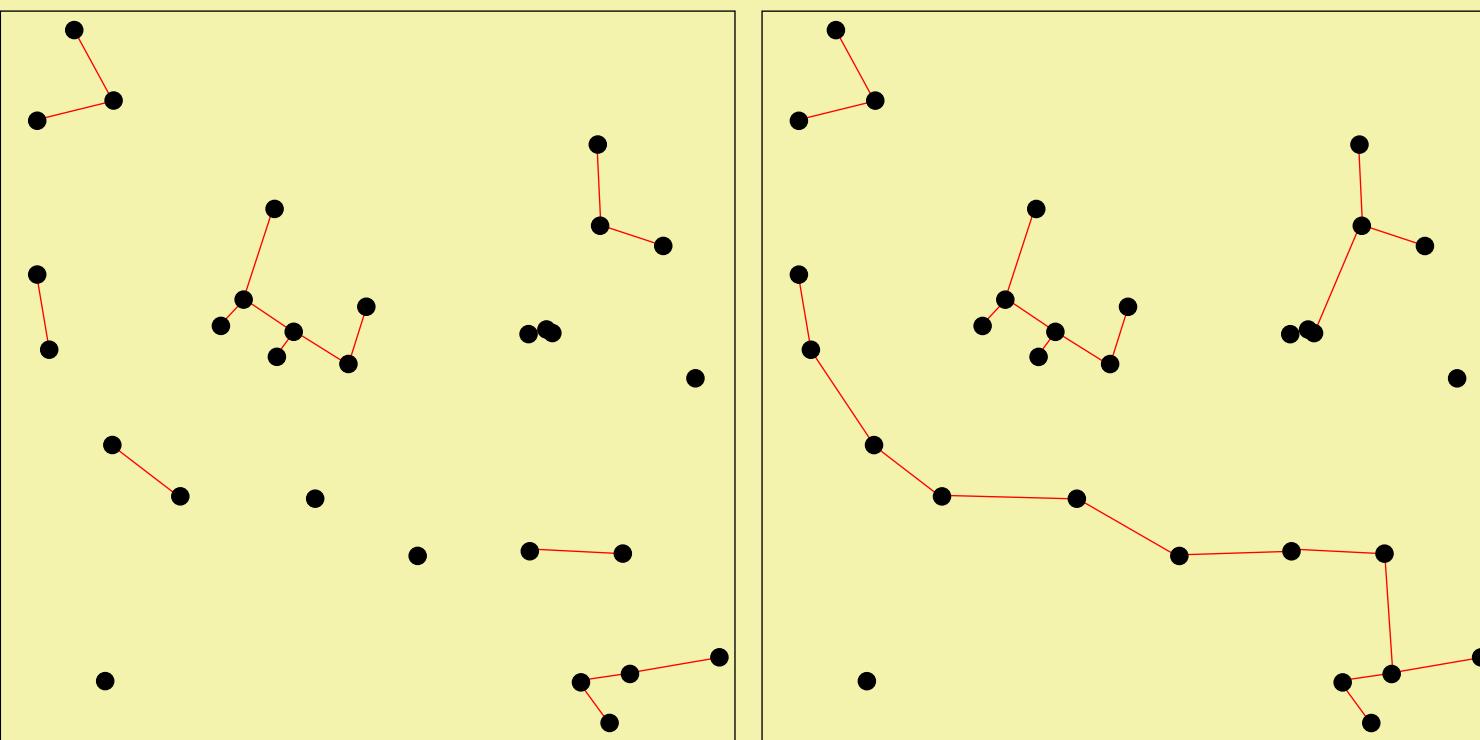
Abstände der Cluster

z.B. Abstand der zwei nächstliegenden Punkte aus den beiden Clustern C_i und C_j , also

$$d_{min}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} |\mathbf{x} - \mathbf{y}|.$$

Damit erhält man den **Nearest Neighbour-Algorithmus**.¹⁷

¹⁷Der Nearest Neighbour-Algorithmus ist nicht zu verwechseln mit der Nearest Neighbour-Methode zur Klassifikation aus Abschnitt 8.2.



- Algorithmus erzeugt minimal aufspannenden Baum¹⁸
- die beiden beschriebenen Algorithmen erzeugen ganz unterschiedliche Cluster
- arbeitet auf Adjazenzmatrix ($O(n^2)$ Zeit und Speicherplatz)
- Schleife wird $n - 1$ mal durchlaufen
- asymptotische Rechenzeit: $O(n^3)$

¹⁸Ein minimal aufspannender Baum ist ein zyklenfreier ungerichteter Graph mit minimaler Summe der Kantenlängen.

Farthest Neighbour-Algorithmus

Abstands von zwei Clustern als Abstand der beiden entferntesten Punkte

$$d_{max}(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} |\mathbf{x} - \mathbf{y}|$$

Alternativ: Abstand der Clustermittelpunkte $d_\mu(C_i, C_j) = |\boldsymbol{\mu}_i - \boldsymbol{\mu}_j|$

andere Clustering-Algorithmen: [Duda/Hart/Stork](#)¹⁹

¹⁹[Duda, R.O./Hart, P.E./Stork, D.G. Pattern Classification. Wiley, 2001.](#)

Data Mining in der Praxis

- Alle vorgestellten Lernverfahren können zum Data Mining verwendet werden.
- teilweise mühsam
- Daten müssen immer passend formatiert werden

Data Mining-System:

- komfortable grafische Benutzeroberfläche
- Werkzeuge zur Visualisierung der Daten
- Vorverarbeitung, wie zum Beispiel die Manipulation von fehlenden Werten
- Analyse der Daten mit Lernverfahren

Das Data Mining Werkzeuge

Das frei verfügbare System KNIME, das wir im folgenden Abschnitt kurz vorstellen werden, bietet eine komfortable Benutzeroberfläche und alle Arten der oben erwähnten Werkzeuge. KNIME verwendet unter anderem WEKA-Module. Außerdem bietet es eine einfache Möglichkeit, mittels eines graphischen Editors den Datenfluss der ausgewählten Visualisierungs-, Vorverarbeitungs- und Analysewerkzeuge zu kontrollieren. Eine ganz ähnliche Funktionalität bietet eine mittlerweile große Zahl anderer Systeme, wie zum Beispiel das Open-Source-Projekt

- RapidMiner (www.rapidminer.com)
- Clementine (www.spss.com/clementine) (baut auf SPSS auf)
- KXEN Analytic Framework (www.kxen.com).
- **KNIME** (Konstanz Information Miner, www.knime.org)

Alternative: Open-Source Java-Programmbibliothek WEKA²⁰

²⁰Witten, I./Frank, E. Data Mining. Hanser Verlag München, 2001.

Das Data Mining Werkzeug KNIME

KNIME

File Edit View Search Node Help

Workflow Pr... Node Repo... *Appendizitis

Misc Weka bayes functions lazy misc trees ADTree Decision Stu... ID3 J48 LMT MSP

File Reader J48 (Weka)

```

graph LR
    FR1[File Reader J48 (Weka)] --> DT[Decision Tree Predictor]
    DT --> S1[Scorer]
    DT --> S2[Scorer]
    S2 --> S3[Scorer]
  
```

Console Node Description J48

Class for generating a j48 decision tree. For further options, click the 'More' - button in the dialog.
The option 'savelInstances' is not recommended for large datasets.

Scorer (#9) - Confusion

App \ Pre...	1	0
1	2952	287
0	861	782

Correct classified: 3734
Wrong classified: 1148
Error: 23.514954 %
Accuracy: 76.4850471118394 %

24M of 40M

J48 (Weka) (#12) - Deci...

File Highlight

Treeview Weka Tree Summary Source

- [root]: class '1' (6585.0 of 9764.0)
 - [Leuko <= 11030]: class '0' (2235.0 of 3460)
 - [SbeiLos <= 0]: class '0' (1412.2226)
 - [TRekt <= 381]: class '0' (1358.0)
 - [LokAbw <= 0]: class '0' (1094.0)
 - [Geschlecht <= 1]: class '1'
 - [Geschlecht > 1]: class '0' (263.5)
 - [LokAbw > 0]: class '1' (81.6789)
 - [TRekt > 381]: class '1' (81.6789)
 - [SbeiLos > 0]: class '1' (1138.647522)
 - [Leuko <= 8600]: class '0' (510.8)
 - [TRekt <= 378]: class '0' (451.0)
 - [TRekt > 378]: class '1' (116.5)
 - [Leuko > 8600]: class '1' (672.77)
 - [Leuko > 11030]: class '1' (4823.0 of 57)
 - [Leuko > 11030]: class '1' (4823.0 of 57)

KNIME

File Edit View Node Search Help

workflow.knime *Appendizitis

```

graph LR
    N1[File Reader] --> N3[Missing Value]
    N3 --> N6[Normalizer]
    N6 --> N8[Node 8]
    N8 --> N9[Scorer]
    N8 --> N7[Node 7]
    N7 --> N10[Node 10]
    
    N2[File Reader] --> N4[Missing Value]
    N4 --> N5[Normalizer (Apply)]
    N5 --> N7
    
    N7 --> N9
    N7 --> N10
  
```

RProp MultiLayerPerceptron Learner

File Reader Missing Value Normalizer

File Reader Missing Value

Scorer

CSV Writer

Console

RProp MultiLayerPerceptron Learner (#8) - Eri

Scorer (#9) - Conf

Correct classified: 2367

Wrong classified: 664

Error: 21.906961 %

Accuracy: 78.09303860112175 %

Default Settings

Zusammenfassung

Lernen mit Lehrer

faules Lernen (lazy learning)

- k-Nearest-Neighbour-Methode (Klass. + Approx.)
- Fallbasiertes Lernen (Klass. + Approx.)

eifriges Lernen (eager learning)

- Induktion von Entscheidungsbäumen (Klass.)
- Lernen von Bayes-Netzen (Klass. + Approx.)
- neuronale Netze (Klass. + Approx.)

Lernen ohne Lehrer (Clustering)

- Nearest Neighbour-Methode
- Farthest Neighbour-Methode
- k-Means
- neuronale Netze

Lernen durch Verstärkung

- Wert-Iteration
- Q-Lernen
- TD-Lernen
- Policy-Gradient-Methoden
- neuronale Netze

Offene Fragen / Forschung

automatische Merkmalsextraktion: (engl. feature selection)

- Kann eine Maschine neue Merkmale finden?
- z.B. mittels Berechnung des Informationsgewinns von Merkmalen
- Clustering zum automatischen kreativen „Entdecken“ von Merkmalen

Literatur

- Mitchell, T. Machine Learning. McGraw Hill, 1997
- Alpaydin, E. Introduction to Machine Learning. MIT Press, 2004
- Duda, R.O./Hart, P.E./Stork, D.G. Pattern Classification. Wiley, 2001
- Journal of Machine Learning Research (<http://jmlr.csail.mit.edu>, frei
verfügbar!)
- Machine Learning Journal
- International Conference on Machine Learning (ICML).
- Machine Learning Repository der Universität Irvine (UCI): D.J. Newman, S. Het-
tich, C.L. Blake/Merz, C.J. UCI Repository of machine learning databases. <http://www.ics.uci.edu/\protect\unhbox\voidb@x\penalty\@M\{\}mlearn/MLRepository.html>, 1998

Kapitel 9

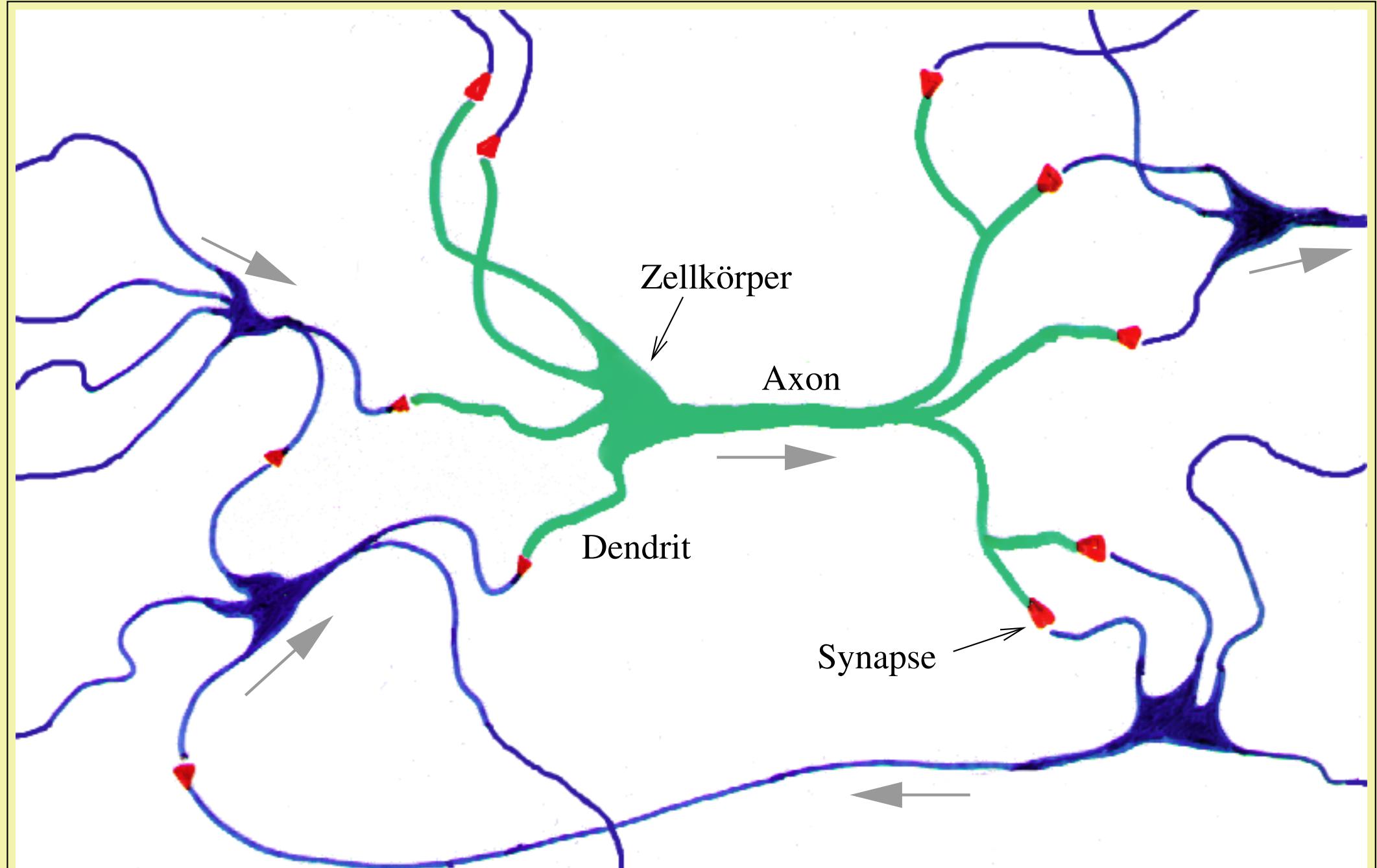
Neuronale Netze

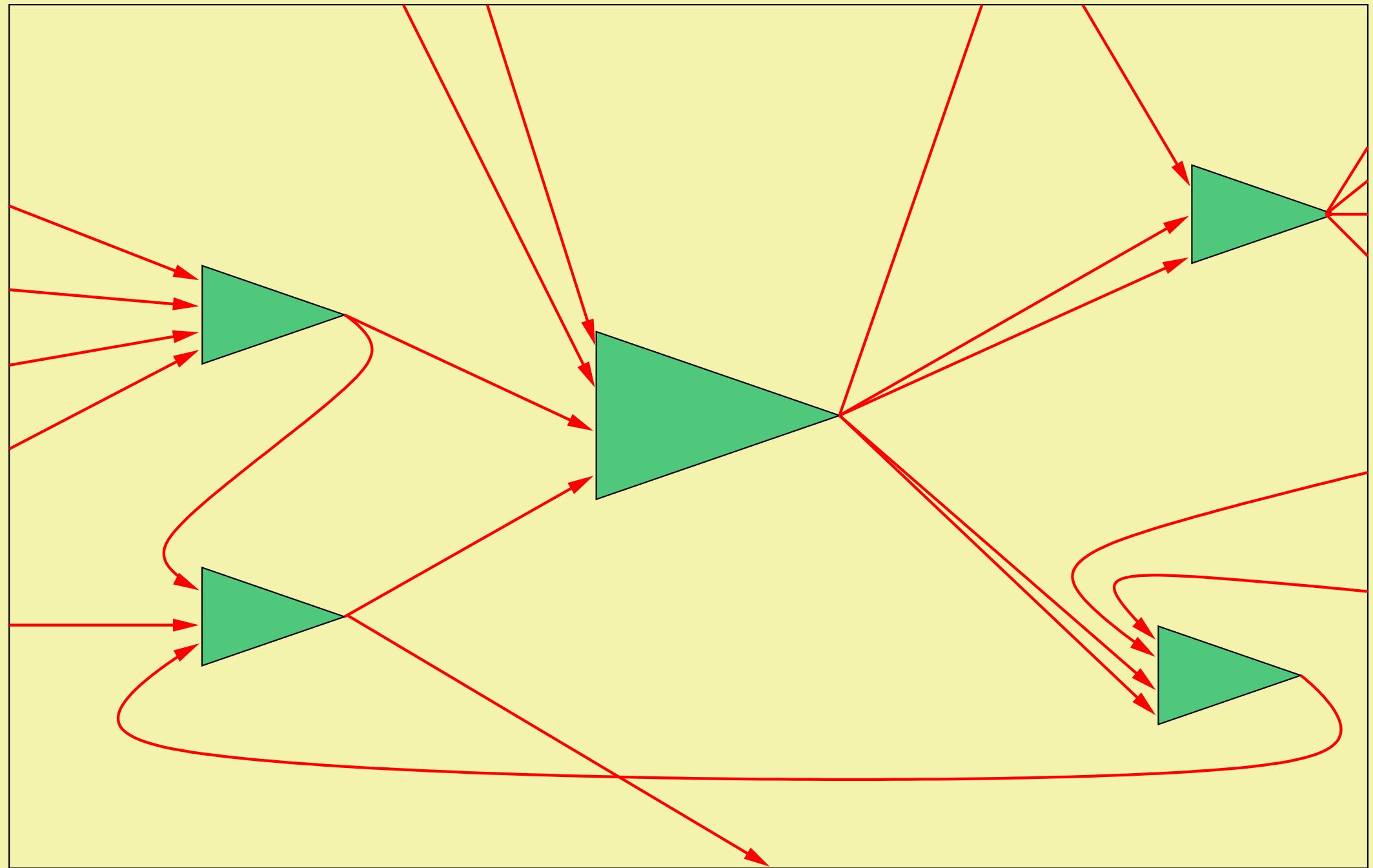
- Etwa 10 bis 100 Milliarden Nervenzellen besitzt das menschliche Gehirn.
- ca. 1900 biologisch bekannt
- 1943 McCulloch und Pitts: „A logical calculus of the ideas immanent in nervous activity“ (in¹).
- Bionik-Zweig innerhalb der KI
- Jedes Neuron ist mit etwa 1000 bis 10 000 anderen Neuronen verbunden

¹Anderson, J./Rosenfeld, E. Neurocomputing: Foundations of Research. Cambridge, MA: MIT Press, 1988.

- ungefähr 10^{14} Verbindungen
- Zeit für einen Impuls eines Neurons beträgt etwa eine Millisekunde
- Taktfrequenz der Neuronen liegt also unter einem Kiloherz

Von der Biologie zur Simulation

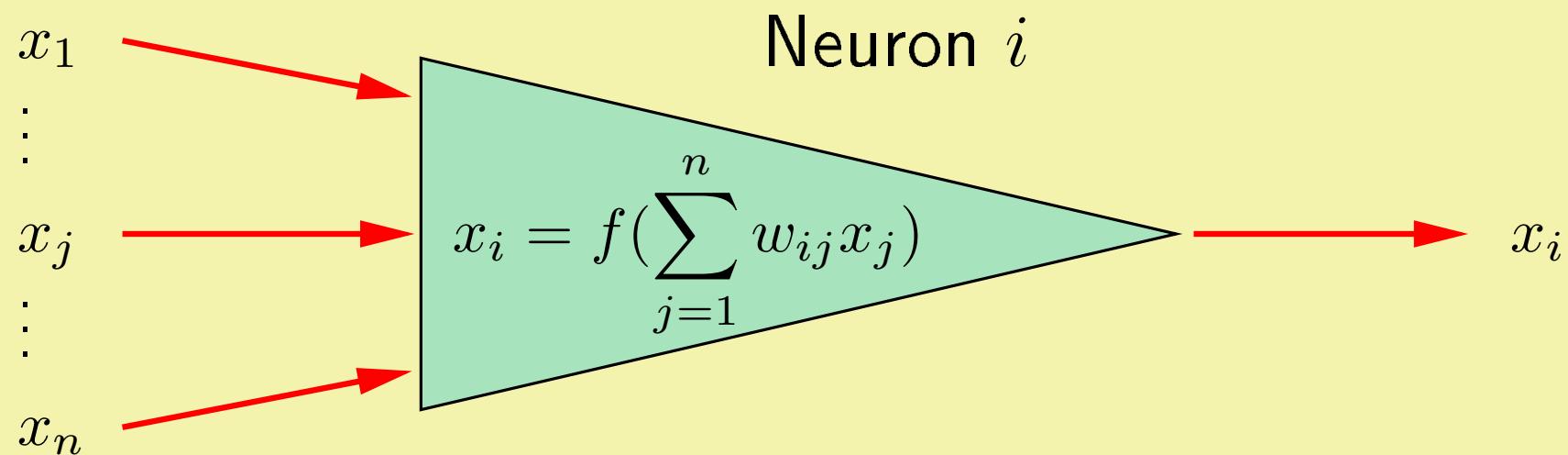




Das mathematische Modell

Das „Aufladen“ des Aktivierungspotentials, durch Summation der gewichteten Ausgabewerte x_1, \dots, x_n aller eingehenden Verbindungen

$$\sum_{j=1}^n w_{ij}x_j.$$



Aktivierungsfunktion f

$$x_i = f \left(\sum_{j=1}^n w_{ij} x_j \right)$$

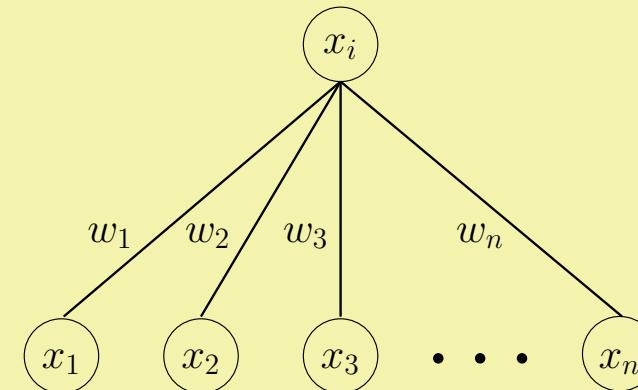
Schwellwertfunktion (Heavisidesche Stufenfunktion)

$$f(x) = H_\Theta(x) = \begin{cases} 0 & \text{falls } x < \Theta \\ 1 & \text{sonst} \end{cases}.$$

Das Neuron berechnet dann seine Ausgabe als

$$x_i = \begin{cases} 0 & \text{falls } \sum_{j=1}^n w_{ij} x_j < \Theta \\ 1 & \text{sonst} \end{cases}.$$

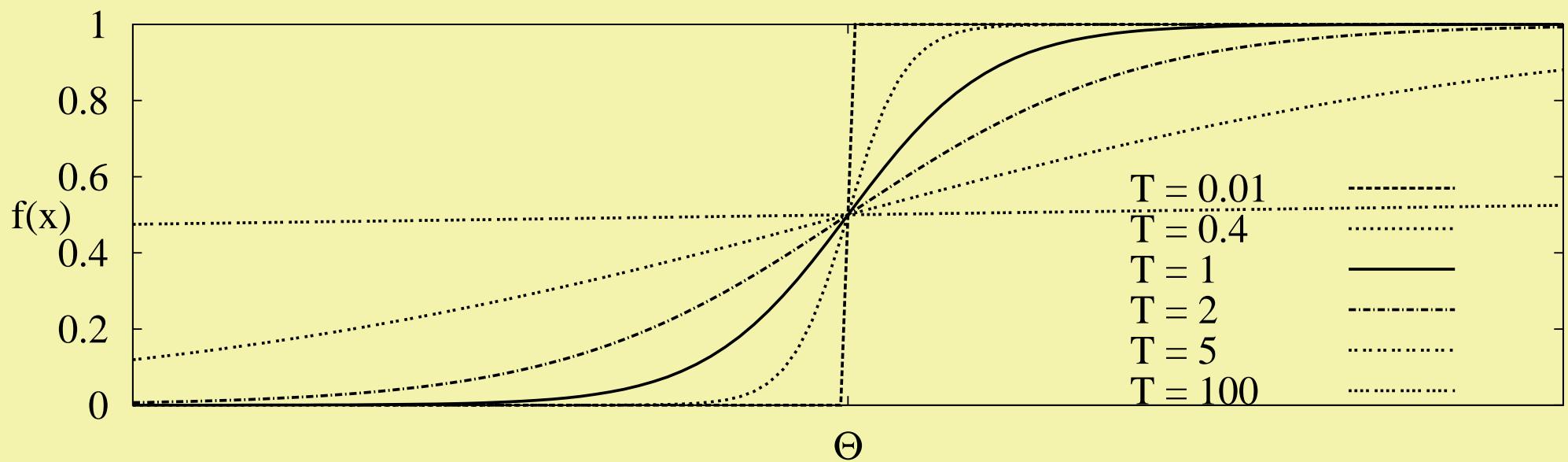
Diese Formel ist identisch mit einem Perzeptron mit der Schwelle Θ .



Stufenfunktion ist unstetig. Alternative:

Sigmoid-Funktion:

$$f(x) = \frac{1}{1 + e^{-\frac{x-\Theta}{T}}}$$



Modellierung des Lernens

D. Hebb, 1949 (**Hebb-Regel**):

Wenn es eine Verbindung w_{ij} von Neuron j und Neuron i gibt und wiederholt Signale von Neuron j zu Neuron i geschickt werden, was dazu führt, dass die beiden Neuronen gleichzeitig aktiv sind, dann wird das Gewicht w_{ij} verstärkt. Eine mögliche Formel für die Gewichtsänderung Δw_{ij} ist

$$\Delta w_{ij} = \eta x_i x_j$$

mit der Konstante η (Lernrate), welche die Größe der einzelnen Lernschritte bestimmt.

Hopfield-Netze

- Bei der Hebb-Regel können die Gewichte im Laufe der Zeit nur wachsen.
- Schwächerwerden ist nicht möglich.
- Dies kann man z.B. durch eine Zerfallskonstante modellieren, die ein unbenutztes Gewicht pro Zeitschritt um einen konstanten Faktor, etwa 0.99, abschwächt.

Andere Lösung (Hopfield 1982)

- binäre Neuronen mit Werten ± 1
- $\Delta w_{ij} = \eta x_i x_j$
- Wann wird Δw_{ij} positiv / negativ ?

Autoassoziativspeicher

- In einem Autoassoziativspeicher können Muster gespeichert werden.
- Abrufen: ähnliches Muster vorgeben.
- klassische Anwendung: Erkennung von Schriftzeichen

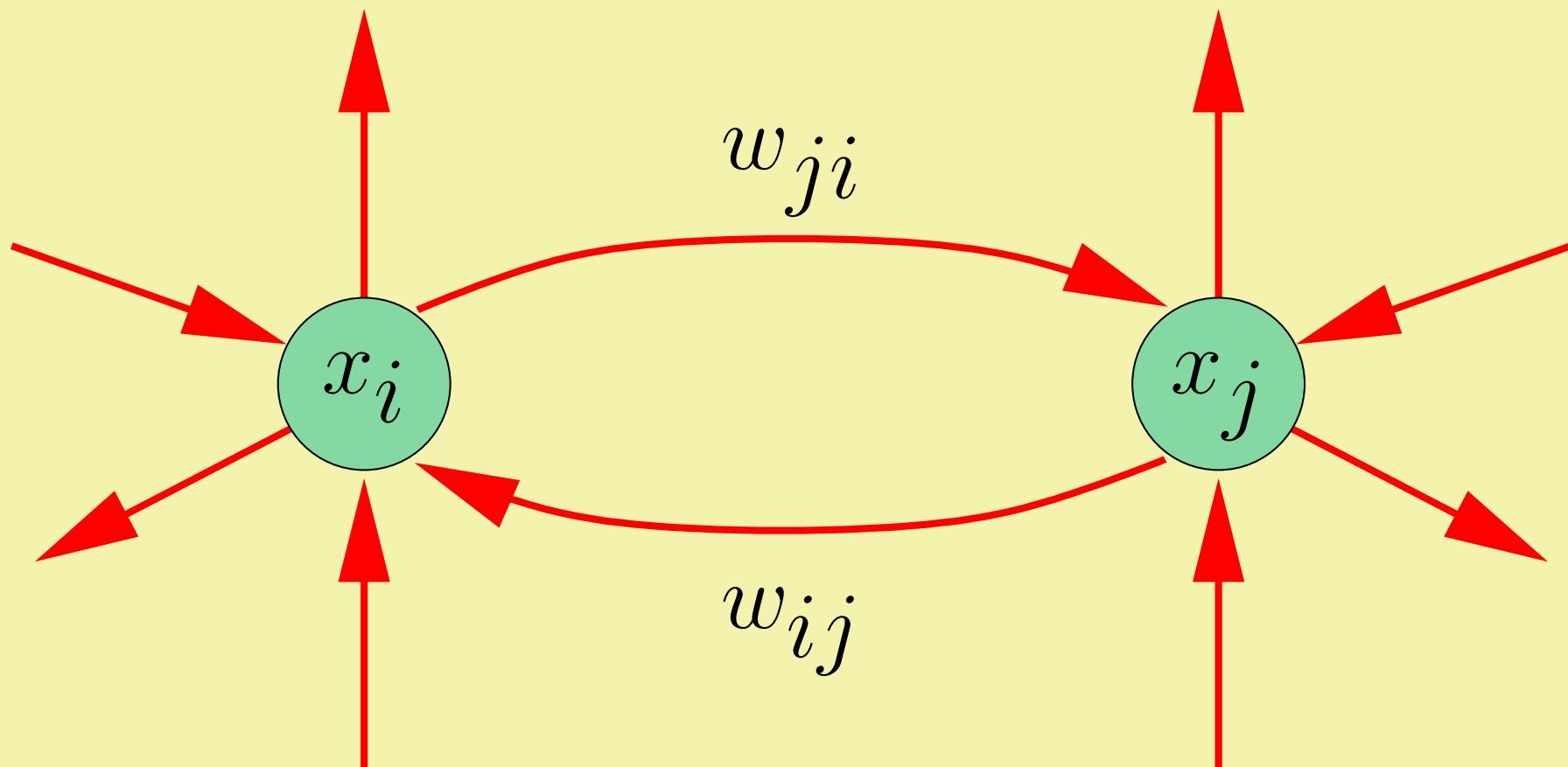
Lernphase:

N binär kodierte Muster $\mathbf{q}^1, \dots, \mathbf{q}^N$ sollen gelernt werden.

$q_i^j \in \{-1, 1\}$: ein Pixel eines Musters.

- n Pixel: neuronales Netz mit n Neuronen
- Neuronen sind vollständig vernetzt
- Gewichtsmatrix symmetrisch
- Diagonalelemente $w_{ii} = 0$

Rekurrente Verbindung von zwei Neuronen



Lernen

$$w_{ij} = \frac{1}{n} \sum_{k=1}^p q_i^k q_j^k. \quad (9.1)$$

Verwandtschaft zur Hebb-Regel!

Mustererkennung

Man legt ein neues Muster x an und aktualisiert alle Neuronen nach

$$x_i = \begin{cases} -1 & \text{falls } \sum_{\substack{j=1 \\ j \neq i}}^n w_{ij} x_j < 0 \\ 1 & \text{sonst} \end{cases} \quad (9.2)$$

bis das Netz stabil wird.

Programmschema

HOPFIELDASSOZIATOR(q)

Initialisiere alle Neuronen: $x = q$

Repeat

$i = \text{Random}(1, n)$

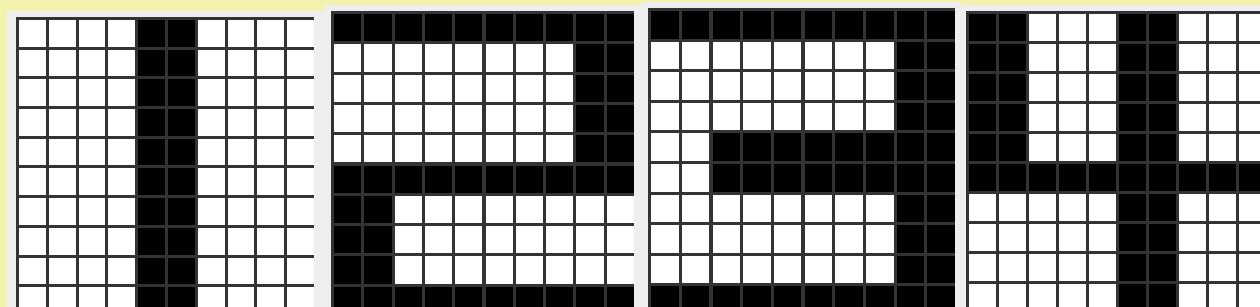
Aktualisiere Neuron i nach Gleichung 9.2

Until x konvergiert

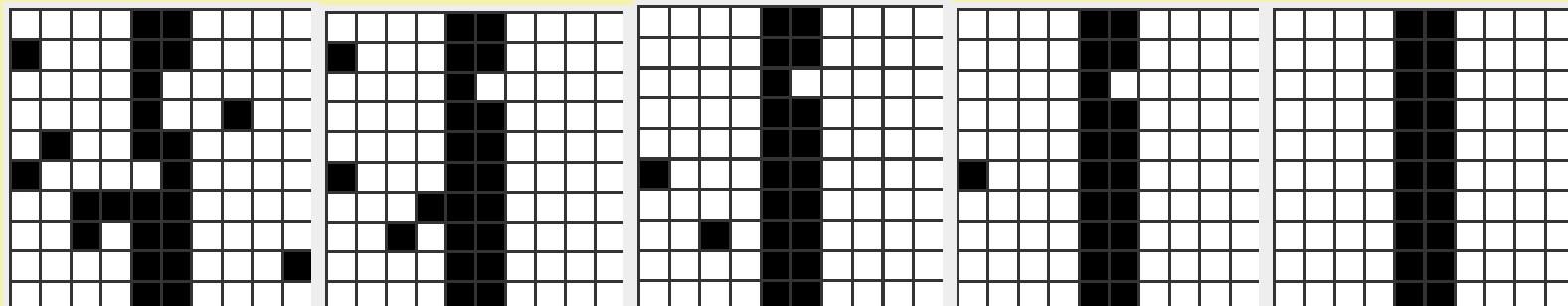
Return(x)

Anwendung auf ein Mustererkennungsbeispiel

- Ziffern in einem 10×10 Pixelfeld
- Hopfield-Netz hat also 100 Neuronen
- insgesamt $\frac{100 \cdot 99}{2} = 4950$ Gewichte



Die vier vom Netz gelernten Trainingsbeispiele.



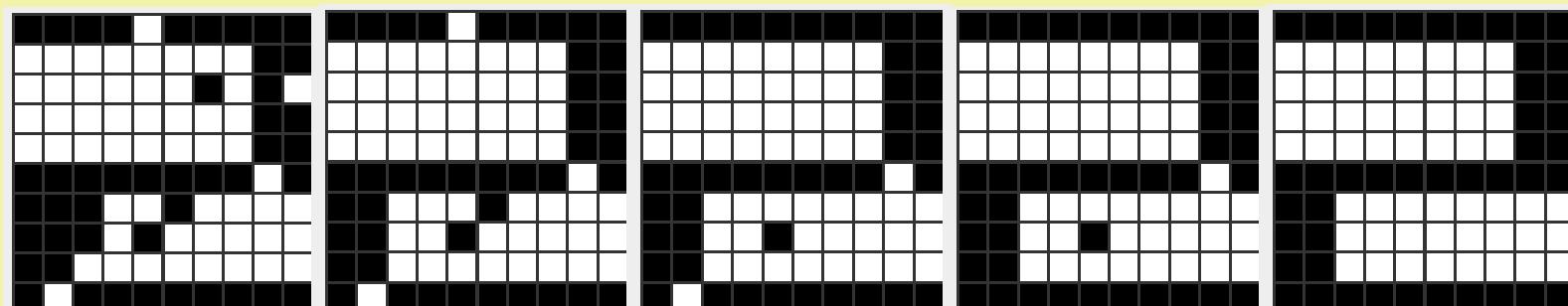
10% Rauschen

62 Schritte

144 Schritte

232 Schritte

379 Schritte



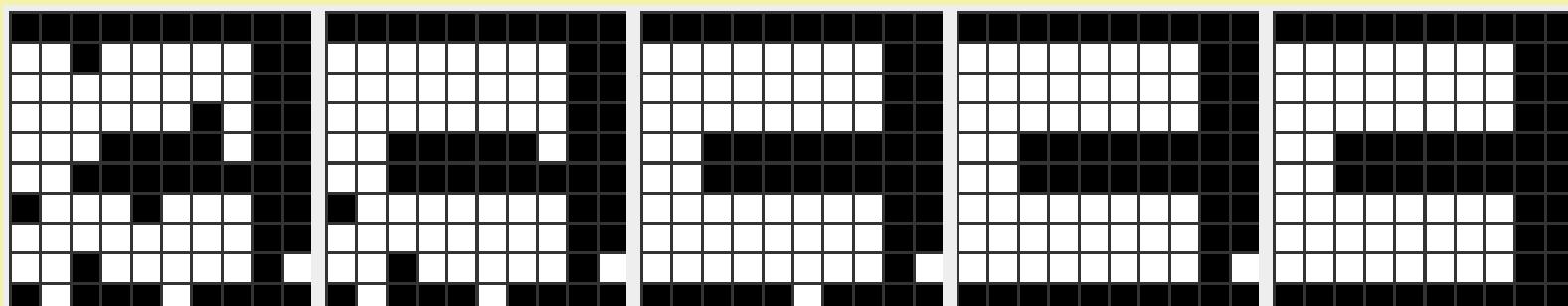
10% Rauschen

65 Schritte

147 Schritte

202 Schritte

301 Schritte



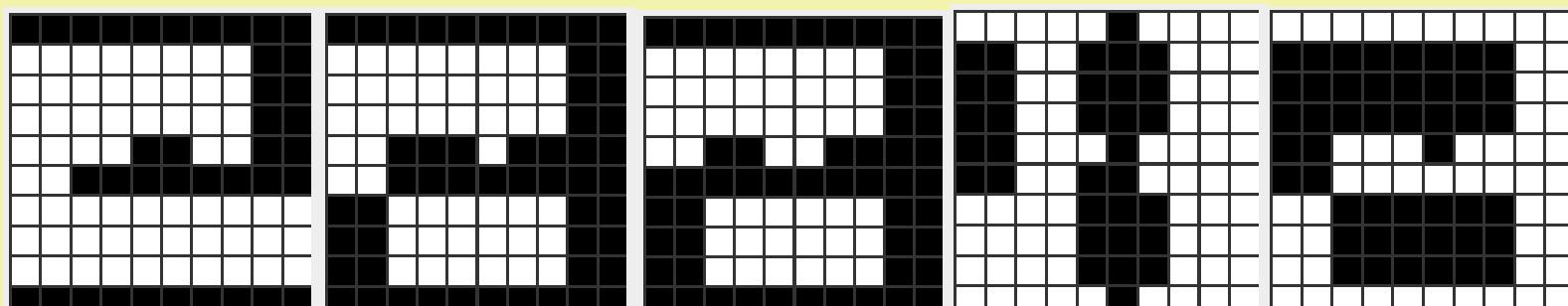
10% Rauschen

66 Schritte

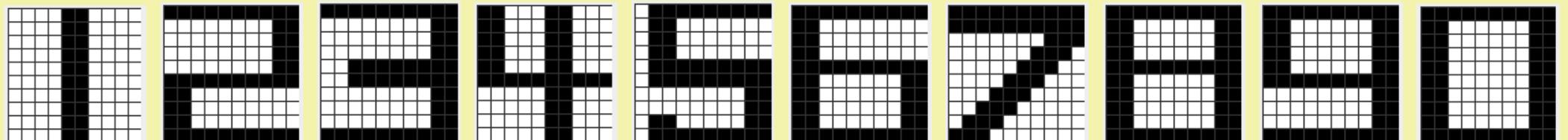
112 Schritte

150 Schritte

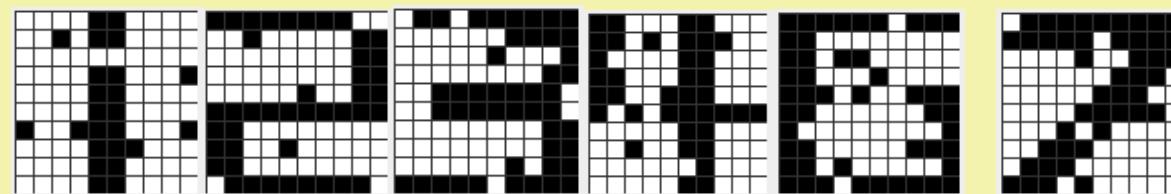
359 Schritte



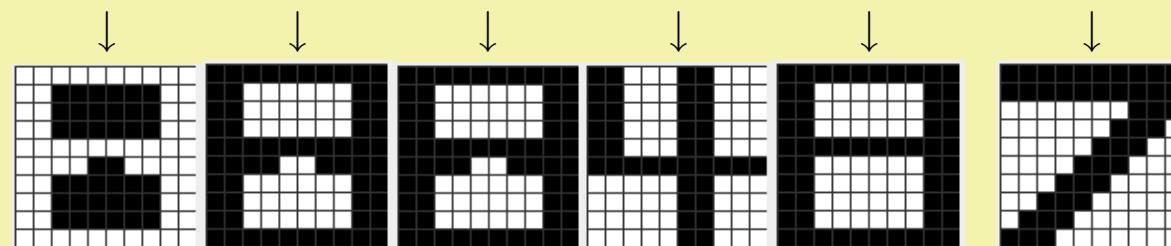
Einige stabile Zustände des Netzes, die nicht gelernt wurden



Die zehn gelernten Muster.



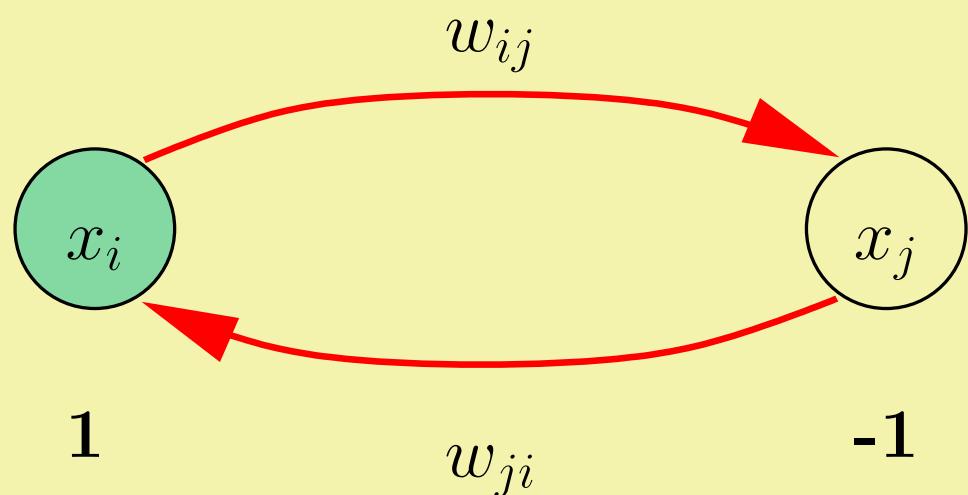
Sechs verrauschte Muster mit etwa 10% Rauschen.



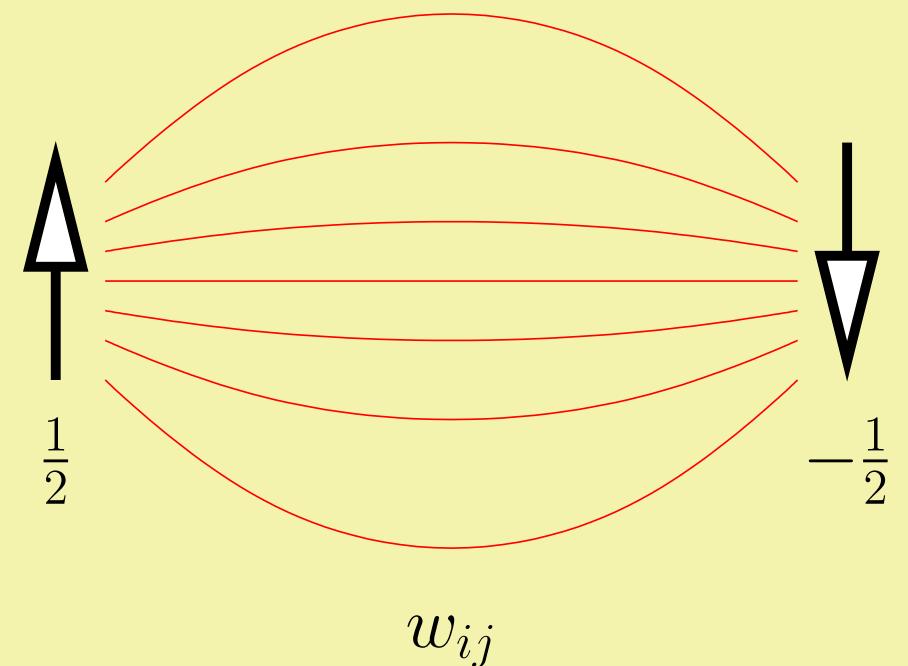
Die aus den verrauschten Mustern entstandenen stabilen Zustände.

Analyse

Neuronen



Elementarmagnete



Neuronale und physikalische Interpretation des Hopfield-Modells.

Gesamtenergie des Systems:

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} x_i x_j.$$

Auch in der Physik gilt $w_{ii} = 0$ und $w_{ij} = w_{ji}$.

Hopfield-Dynamik

- physikalisches System im Gleichgewicht minimiert $E(\mathbf{x}, \mathbf{y})$.
- System bewegt sich auf einen Zustand minimaler Energie zu.
- Hopfield-Dynamik aktualisiert in jedem Schritt den Zustand eines Neurons so, dass es den mit kleinerer Gesamtenergie einnimmt.

Beitrag des Neurons i zur Gesamtenergie:

$$-\frac{1}{2}x_i \sum_{j \neq i}^n w_{ij}x_j.$$

Wenn

$$\sum_{j \neq i}^n w_{ij}x_j < 0$$

ist, dann führt $x_i = -1$ zu einem negativen Beitrag zur Gesamtenergie, $x_i = 1$ hingegen zu einem positiven Beitrag.

Analog: Für

$$\sum_{j \neq i}^n w_{ij} x_j \geq 0$$

muss $x_i = 1$ sein.

- gesamte Energie des Systems nimmt im Laufe der Zeit monoton ab.
- Netzwerk bewegt sich auf einen Zustand minimaler Energie zu.
- Welche Bedeutung haben diese Minima der Energiefunktion?

Analog: Für

$$\sum_{j \neq i}^n w_{ij} x_j \geq 0$$

muss $x_i = 1$ sein.

- gesamte Energie des Systems nimmt im Laufe der Zeit monoton ab.
- Netzwerk bewegt sich auf einen Zustand minimaler Energie zu.
- Welche Bedeutung haben diese Minima der Energiefunktion?
- Die gelernten Muster stellen Minima der Energiefunktion dar.
- Falls jedoch zu viele Muster gelernt werden, so konvergiert das System gegen Minima, die keinen gelernten Mustern entsprechen.
- Wechsel von Ordnung zu Chaos.
- Phasenübergang

Physik

- Schmelzen eines Eiskristalls.
- Im Kristall herrscht ein Zustand hoher Ordnung.
- Im flüssigen Wasser ist die Struktur der Moleküle aufgelöst.

Neuronales Netz

- Phasenübergang
- geordneten Lernen und Wiedererkennen von Mustern
- chaotisches Lernen im Fall von zu vielen Mustern.
- Effekten, die wir bei uns selbst gelegentlich beobachten.

Phasenübergang im Hopfield-Netz

alle Neuronen seien in Musterzustand q^1

$\ln \sum_{\substack{j=1 \\ j \neq i}}^n w_{ij} q_j$ werden die gelernten Gewichte aus Gleichung 9.1 eingesetzt:

$$\begin{aligned} \sum_{\substack{j=1 \\ j \neq i}}^n w_{ij} q_j^1 &= \frac{1}{n} \sum_{\substack{j=1 \\ j \neq i}}^n \sum_{k=1}^N q_i^k q_j^k q_j^1 = \frac{1}{n} \sum_{\substack{j=1 \\ j \neq i}}^n \left(q_i^1 (q_j^1)^2 + \sum_{k=2}^N q_i^k q_j^k q_j^1 \right) \\ &= q_i^1 + \frac{1}{n} \sum_{\substack{j=1 \\ j \neq i}}^n \sum_{k=2}^N q_i^k q_j^k q_j^1 \end{aligned}$$

i -te Komponente des angelegten Musters plus eine Summe mit $(n - 1)(N - 1)$ Summanden. Wenn diese Summanden alle statistisch unabhängig sind, kann man

die Summe durch eine normalverteilte Zufallsvariable mit Standardabweichung

$$\frac{1}{n} \sqrt{(n-1)(N-1)} \approx \sqrt{\frac{N-1}{n}}$$

beschreiben.

- Rauschen nicht störend, solange $N \ll n!$
- genauere Berechnung ergibt als kritischen Punkt $N = 0.146 n$.
- Beispiel: bei 100 Neuronen können bis zu 14 unkorrelierte Muster gespeichert werden.
- Speicherkapazität weit unter der von Listenspeichern!
- Hopfield-Netze arbeiten nur dann gut, wenn Muster mit etwa 50% 1-Bits gelernt werden.
- Abhilfe: Neuronen mit Schwelle

Zusammenfassung und Ausblick

- Hopfield Modell trug in den achtziger Jahren zur Welle der Begeisterung für neuronale Netze bei
- Neuroinformatik
- Hopfield-Netze sind rekurrent
- Netze ohne Rückkopplungen sind einfacher zu verstehen
- Problem: lokale Minima \Rightarrow Boltzmann-Maschine, „simulated annealing“
- Hopfield-Dynamik auch auf andere Energiefunktionen anwendbar
- z.B. Problem des Handlungsreisenden

Neuronale Assoziativspeicher

- Telefonbuch: Abbildung von Namen auf Telefonnummer
- realisiert über Tabelle in Datenbank
- Zugangskontrolle zu einem Gebäude mittels Foto des Gesichts der zu kontrollierenden Person
- Probleme, wenn nur Fotos in Datenbank gespeichert werden
- Lösung: **Assoziativspeicher:** kann auch „ähnliche“ Fotos dem richtigen Namen zuordnen.
- typische Aufgabe für maschinelle Lernverfahren.
- Einfacher Ansatz: Nearest Neighbour Methode.
- Für Zugangskontrolle aber nicht anwendbar. Warum?

Korrelationsmatrix-Speicher

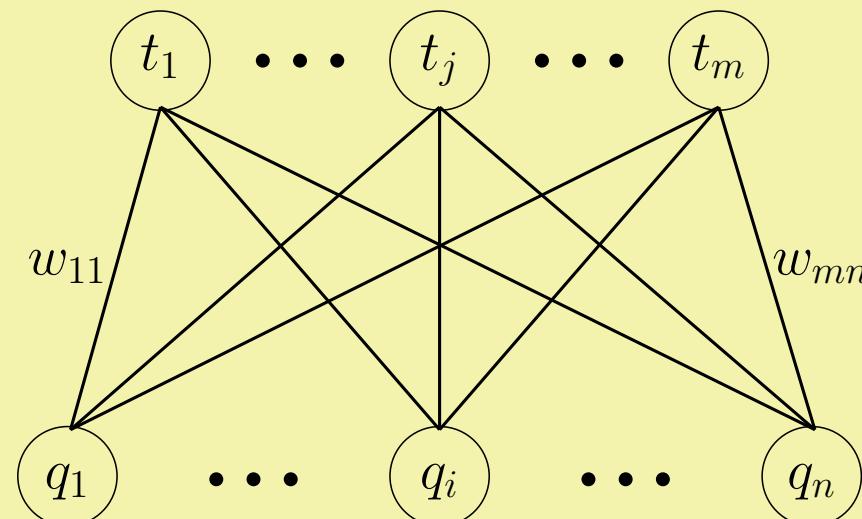
gegeben: Trainingsdaten mit N Anfrage-Antwort-Paaren:

$$T = \{(\mathbf{q}^1, \mathbf{t}^1), \dots, (\mathbf{q}^N, \mathbf{t}^N)\}$$

gesucht: Matrix \mathbf{W} , die alle Anfragevektoren korrekt auf ihre Antworten abbildet.

für $p = 1, \dots, N$

$$\mathbf{t}^p = \mathbf{W} \cdot \mathbf{q}^p, \quad \text{bzw.} \quad t_i^p = \sum_{j=1}^n w_{ij} q_j^p \quad (9.3)$$



Berechnung der Matrixelemente w_{ij} (Hebb-Regel):

$$w_{ij} = \sum_{p=1}^N q_j^p t_i^p \quad (9.4)$$

Definition 9.2 Zwei Vektoren \mathbf{x} und \mathbf{y} heißen *orthonormal*, wenn

$$\mathbf{x} \cdot \mathbf{y} = \begin{cases} 1 & \text{falls } \mathbf{x} = \mathbf{y} \\ 0 & \text{sonst} \end{cases}$$

Satz 9.2 Wenn alle N Anfragevektoren \mathbf{q}^p in den Trainingsdaten orthonormal sind, so wird jeder Vektor \mathbf{q}^p durch Multiplikation mit der Matrix w_{ij} aus Gleichung 9.4 auf den entsprechenden Antwortvektor \mathbf{t}^p abgebildet.

Beweis: Wir setzen Gleichung 9.4 in Gleichung 9.3 ein und erhalten

$$\begin{aligned}
 (\mathbf{W} \cdot \mathbf{q}^p)_i &= \sum_{j=1}^n w_{ij} q_j^p = \sum_{j=1}^n \sum_{r=1}^N q_j^r t_i^r q_j^p = \sum_{j=1}^n \left(q_j^p q_j^p t_i^p + \sum_{\substack{r=1 \\ r \neq p}}^N q_j^r q_j^p t_i^r \right) \\
 &= t_i^p \underbrace{\sum_{j=1}^n q_j^p q_j^p}_{=1} + \sum_{\substack{r=1 \\ r \neq p}}^N t_i^r \underbrace{\sum_{j=1}^n q_j^r q_j^p}_{=0} = t_i^p
 \end{aligned}$$

□

Problem: sei der Name Hans einem Gesicht zugeordnet,

Ausgabe bei Eingabe eines ähnlichen Gesichtes:

z.B. „Gans“ oder „Hbns“

Die Pseudoinverse

Anfragevektoren als Spalten einer $n \times N$ Matrix: $\mathbf{Q} = (\mathbf{q}^1, \dots, \mathbf{q}^N)$
 Antwortvektoren als Spalten einer $m \times N$ Matrix $\mathbf{T} = (\mathbf{t}^1, \dots, \mathbf{t}^N)$:

$$\mathbf{T} = \mathbf{W} \cdot \mathbf{Q} \quad (9.5)$$

Gleichung nach \mathbf{W} aufzulösen:

$$\mathbf{W} = \mathbf{T} \cdot \mathbf{Q}^{-1}. \quad (9.6)$$

Wie soll man eine nicht invertierbare Matrix invertieren?

Eine Matrix \mathbf{Q} ist invertierbar genau dann, wenn es eine Matrix \mathbf{Q}^{-1} gibt mit der Eigenschaft

$$\mathbf{Q} \cdot \mathbf{Q}^{-1} = \mathbf{I}, \quad (9.7)$$

Gesucht: Matrix, die dieser Eigenschaft am nächsten kommt

Definition 9.4 Sei \mathbf{Q} eine reelle $n \times m$ Matrix. Eine $m \times n$ Matrix \mathbf{Q}^+ heißt **Pseudoinverse** zu \mathbf{Q} , wenn sie

$$\|\mathbf{Q} \cdot \mathbf{Q}^+ - \mathbf{I}\|$$

minimiert. Hierbei ist $\|M\|$ die quadratische Norm, das heißt die Summe der Quadrate aller Matrixelemente von M .

Nun:

$$\mathbf{W} = \mathbf{T} \cdot \mathbf{Q}^+ \quad (9.8)$$

Gewichtsmatrix \mathbf{W} minimiert den Assoziationsfehler (engl. crosstalk) $\mathbf{T} - \mathbf{W} \cdot \mathbf{Q}$

- Das Finden der Pseudoinversen ist nicht einfach.
- Backpropagation-Algorithmus

Die binäre Hebb-Regel (Palm-Modell)

Muster sind binär kodiert: $q^p \in \{0, 1\}^n$ und $t^p \in \{0, 1\}^m$.

$n = 10, m = 6$:

q^3				1	1		1			
q^2	1	1						1		
q^1		1				1		1		
	1	1						1		1
		1				1		1		
	1	1	1	1		1	1	1	1	1
			1	1		1				1

$t^1 \ t^2 \ t^3$

Abrufen der gespeicherten Muster

q^3				1	1		1			
q^2		1	1					1		
q^1			1			1		1		
	1	1					1		2	3
									0	0
		1			1		1		3	2
	1	1	1	1		1	1		3	3
			1	1		1			1	0
									0	0

Berechnung der Produkte Wq^1 , Wq^2 , Wq^3 .

binäre Hebb-Regel: $w_{ij} = \bigvee_{p=1}^N q_j^p t_i^p.$ (9.9)

Speicherkapazität

- Gewichtsmatrix muss dünn besetzt sein

Matrix hat $m n$ Elemente. Ein zu speicherndes Paar hat $m + n$ Bits.

Zahl der speicherbaren Muster N_{max} :²

$$\alpha = \frac{\text{Zahl speicherbarer Bits}}{\text{Zahl der Speicherzellen}} = \frac{(m + n)N_{max}}{m n} \leq \ln 2 \approx 0.69. \quad (9.10)$$

Speichermodell	α
Listenspeicher	1
Der Assoziativspeicher mit binärer Hebb-Regel	0.69
Kohonen-Assoziativspeicher	0.72
Hopfield-Netz	0.292

²Palm, G. On Associative Memory. Biological Cybernetics, 36 1980.

Ein Fehlerkorrekturprogramm

Paarkodierung für Anfragevektoren q

Anfragevektor besitzt 676 Bits für jedes der Paare

aa, ab, . . . , az, ba, . . . , bz, . . . , za, . . . , zz.

Beispiel: für „hans“ werden „ha“, „an“ und „ns“ mit Eins besetzt.

$26 \cdot 26 = 676$ geordnete Paare von Buchstaben.

Im **Antwortvektor** t werden für jede Position in dem Wort 26 Bit reserviert (Wortlänge max. 10).

Für „hans“ werden die Bits 8, 27, 66 und 97 besetzt

Die Gewichtsmatrix W hat eine Größe von $676 \cdot 260$ Bit = 199420 Bit.

Speicherkapazität

$$N_{max} \leq 0.69 \frac{m n}{m + n} = 0.69 \frac{676 \cdot 260}{676 + 260} \approx 130 \text{ Worte}$$

Gespeicherte Namen:

agathe, agnes, alexander, andreas, andree, anna, annemarie, astrid, august, bernhard, bjorn, cathrin, christian, christoph, corinna, corrado, dieter, elisabeth, elvira, erdmut, ernst, evelyn, fabrizio, frank, franz, geoffrey, georg, gerhard, hannelore, harry, herbert, ingilt, irmgard, jan, johannes, johnny, juergen, karin, klaus, ludwig, luise, manfred, maria, mark, markus, marleen, martin, matthias, norbert, otto, patricia, peter, phillip, quit, reinhold, renate, robert, robin, sabine, sebastian, stefan, stephan, sylvie, ulrich, ulrike, ute, uwe, werner, wolfgang, xavier

Assoziationen des Programms:

Gib Muster: harry

Schwelle: 4, Antwort: harry

Schwelle: 3, Antwort: harry

Schwelle: 2, Antwort: horryrrde

Gib Muster: ute

Schwelle: 2, Antwort: ute

Gib Muster: gerhar

Schwelle: 5, Antwort: gerhard

Schwelle: 4, Antwort: gerrarn

Gib Muster: andrees

Schwelle: 6, Antwort: a

Schwelle: 5, Antwort: andree

Schwelle: 4, Antwort: andrees

Schwelle: 3, Antwort: mnnrens

Schwelle: 2, Antwort: morxsnssr

Gib Muster: johanne

Schwelle: 6, Antwort: johnnnes

Schwelle: 5, Antwort: johnnnes

Schwelle: 4, Antwort: jornnnrse

Schwelle: 3, Antwort: sorrnyrse

Schwelle: 2, Antwort: wtrrsyrs

Lineare Netze mit minimalem Fehler

Idee: Lernen aus Fehlern

Gegeben: Trainingsvektoren

$$T = \{(\mathbf{q}^1, \mathbf{t}^1), \dots, (\mathbf{q}^N, \mathbf{t}^N)\}$$

mit $\mathbf{q}^p \in [0, 1]^n$ $\mathbf{t}^p \in [0, 1]^m$.

Gesucht: ist eine Funktion $f : [0, 1]^n \rightarrow [0, 1]^m$, welche den mittleren quadratischen Fehler

$$\sum_{p=1}^N (f(\mathbf{q}^p) - \mathbf{t}^p)^2$$

auf den Daten minimiert.

Lösungsvorschlag:

$$f(\mathbf{q}) = 0, \quad \text{falls} \quad \mathbf{q} \notin \{\mathbf{q}^1, \dots, \mathbf{q}^N\}$$

und

$$f(\mathbf{q}^p) = \mathbf{t}^p \quad \forall p \in \{1, \dots, N\}.$$

Warum werden wir mit dieser Funktion nicht glücklich?

Lösungsvorschlag:

$$f(\mathbf{q}) = 0, \quad \text{falls} \quad \mathbf{q} \notin \{\mathbf{q}^1, \dots, \mathbf{q}^N\}$$

und

$$f(\mathbf{q}^p) = \mathbf{t}^p \quad \forall p \in \{1, \dots, N\}.$$

Warum werden wir mit dieser Funktion nicht glücklich?

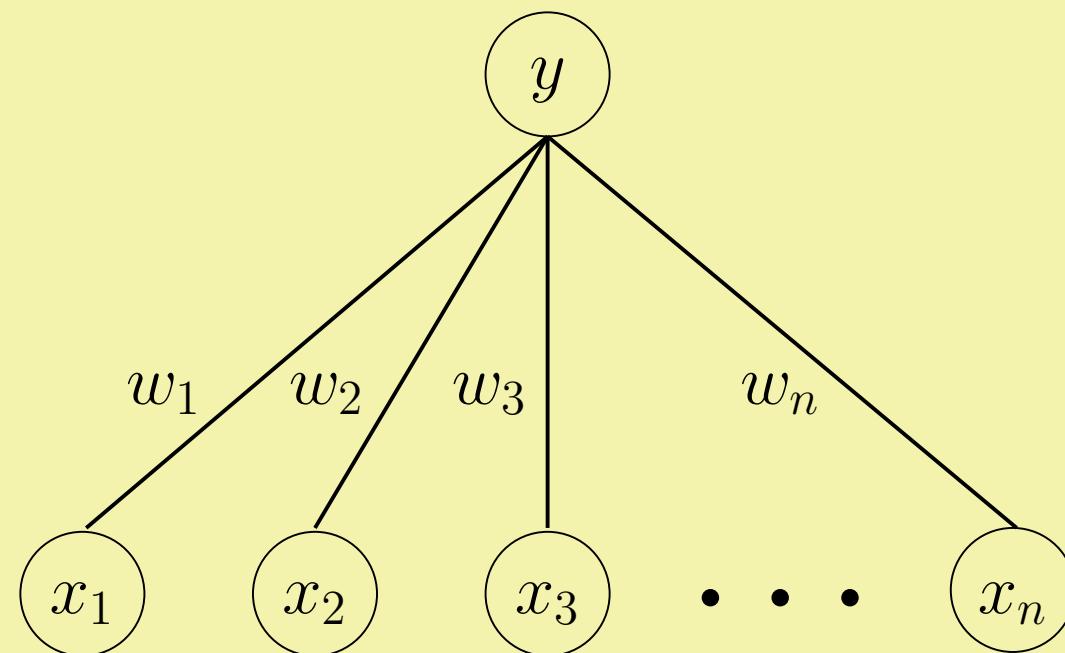
- Weil wir ein intelligentes System bauen wollen!
- Generalisierung von den Trainingsdaten auf neue unbekannte Daten
- Wir wollen keine Überanpassung (engl. overfitting)
- Wir wollen eine Funktion haben, die glatt ist und zwischen den Punkten „ausgleicht“.
- Wir müssen die Funktionenklasse aber noch weiter einschränken.

Die Methode der kleinsten Quadrate

Lineares zweilagiges Netzwerk

$$y = f \left(\sum_{i=1}^n w_i x_i \right)$$

mit $f(x) = x$ berechnet.



Sigmoid-Funktion bringt nichts, denn sie ist streng monoton wachsend!

Gesucht: Vektor \mathbf{w} , der

$$E(\mathbf{w}) = \sum_{p=1}^N (\mathbf{w} \mathbf{q}^p - t^p)^2 = \sum_{p=1}^N \left(\sum_{i=1}^n w_i q_i^p - t^p \right)^2$$

minimiert.

Notwendige Bedingung für ein Minimum der Fehlerfunktion: für $j = 1, \dots, n$ muss

$$\frac{\partial E}{\partial w_j} = 2 \sum_{p=1}^N \left(\sum_{i=1}^n w_i q_i^p - t^p \right) q_j^p = 0$$

sein. Ausmultiplizieren ergibt

$$\sum_{p=1}^N \left(\sum_{i=1}^n w_i q_i^p q_j^p - t^p q_j^p \right) = 0$$

Vertauschen der Summen ergibt LGS:

$$\sum_{i=1}^n w_i \sum_{p=1}^N q_i^p q_j^p = \sum_{p=1}^N t^p q_j^p,$$

Mit

$$A_{ij} = \sum_{p=1}^N q_i^p q_j^p \quad \text{und} \quad b_j = \sum_{p=1}^N t^p q_j^p \quad \text{für } i, j = 1, \dots, n \quad (9.11)$$

als Matrixgleichung (Normalgleichungen):

$$\mathbf{A}\mathbf{w} = \mathbf{b} \quad (9.12)$$

- Normalgleichungen besitzen immer mindestens eine Lösung und wenn \mathbf{A} invertierbar ist, genau eine.
- \mathbf{A} ist positiv definit, was zur Folge hat, dass die gefundene Lösung im eindeutigen Fall ein globales Minimum darstellt.

Rechenzeiten:

- Aufstellen der Matrix: $\Theta(N \cdot n^2)$
- Auflösen des Gleichungssystems: $O(n^3)$.

Diese Methode lässt sich leicht auf mehrere Ausgabeneuronen erweitern

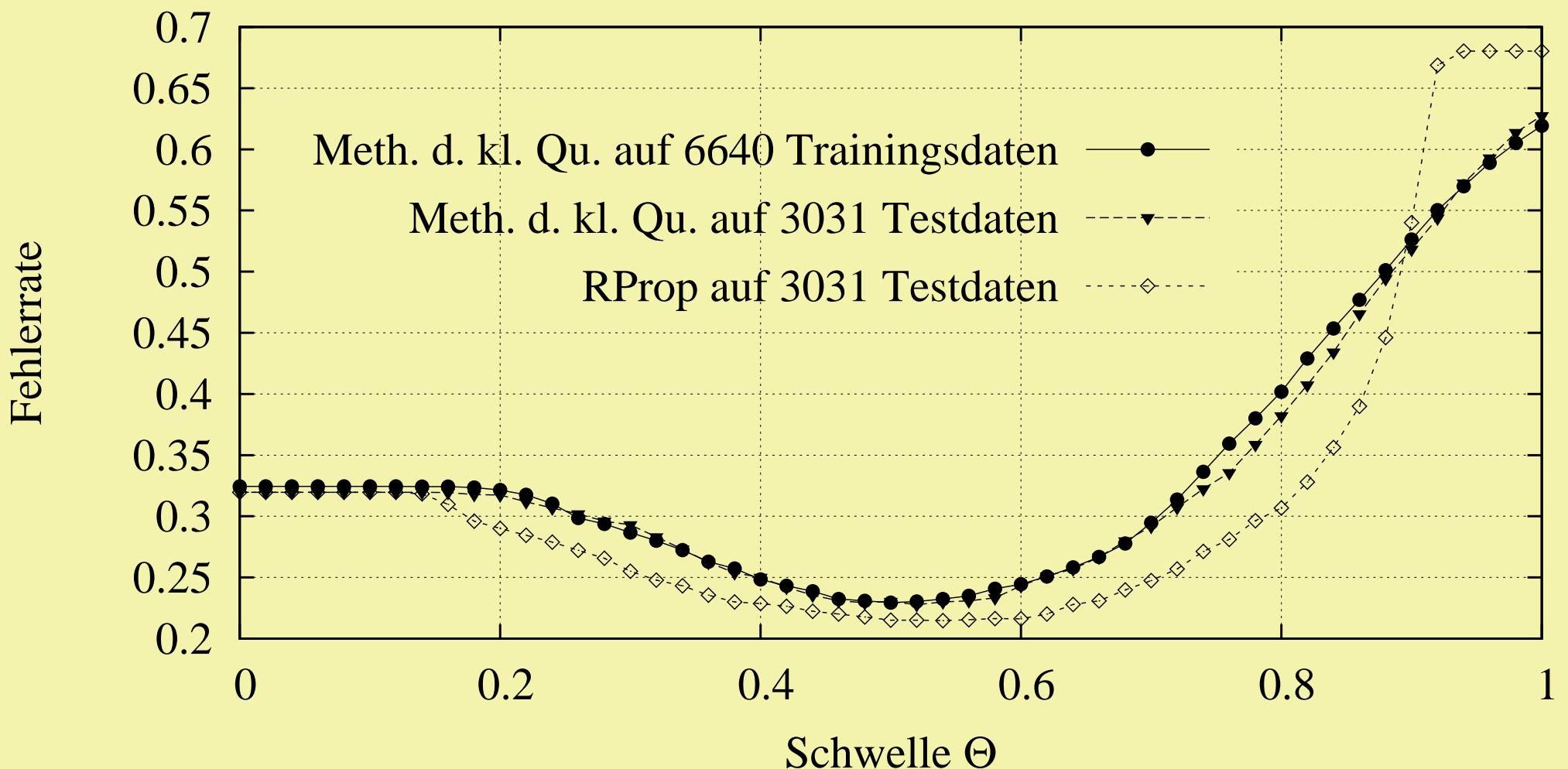
Anwendung auf die Appendizitisdaten

lineare Abbildung von den Symptomen auf die stetige Klassenvariable *AppScore*:

$$\begin{aligned} \textit{AppScore} = & 0.00085 \textit{Alter} - 0.125 \textit{Geschl} + 0.025 \textit{S1Q} + 0.035 \textit{S2Q} - 0.021 \textit{S3Q} \\ & - 0.025 \textit{S4Q} + 0.12 \textit{AbwLok} + 0.031 \textit{AbwGlo} + 0.13 \textit{Losl} + 0.081 \textit{Ersch} \\ & + 0.0034 \textit{RektS} + 0.0027 \textit{TAxi} + 0.0031 \textit{TRek} + 0.000021 \textit{Leuko} \\ & - 0.11 \textit{Diab} - 1.83. \end{aligned}$$

stetige Werte für *AppScore*, obwohl *App* die Werte 0 und 1 annimmt!

Schwellwertentscheidung!



Fehler der Methode der kleinsten Quadrate auf den Trainings- und Testdaten.

Die Delta-Regel

bisher: **Batch-Lernen** jetzt: **inkrementellen Lernen.**

in jedem Schritt werden für ein neues Trainingsbeispiel die Gewichte verändert:

$$w_j = w_j + \Delta w_j$$

Fehlerminimierung, also nochmal partielle Ableitungen der Fehlerfunktion

$$\frac{\partial E}{\partial w_j} = 2 \sum_{p=1}^N \left(\sum_{i=1}^n w_i q_i^p - t^p \right) q_j^p.$$

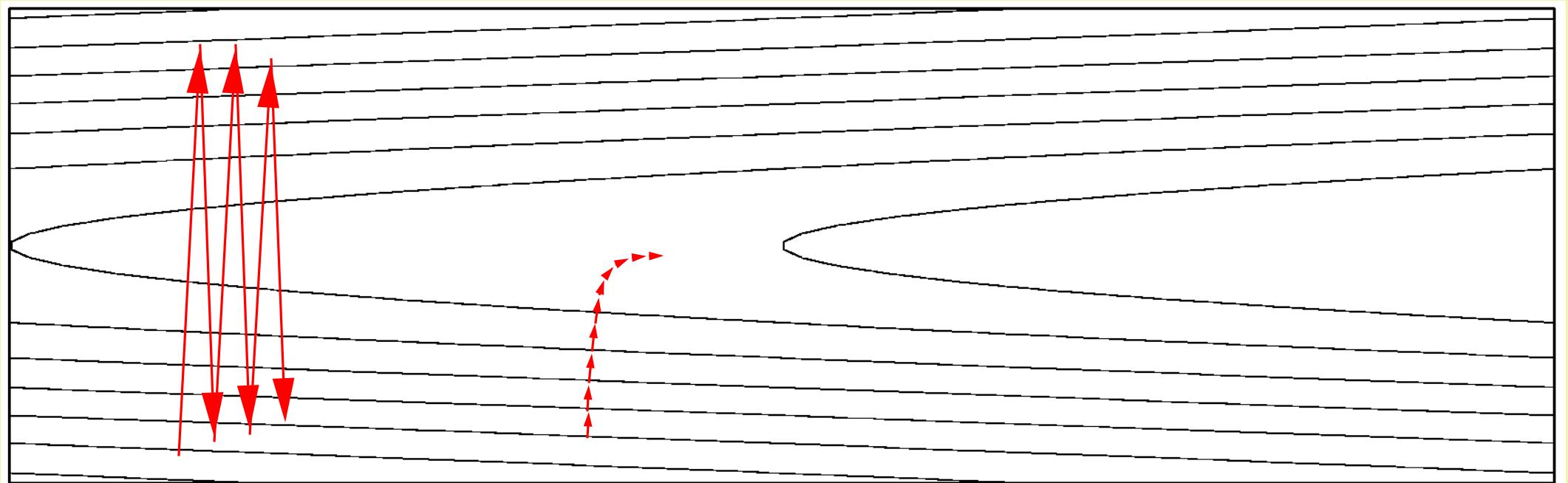
Gradient

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right)$$

Gradientanabstieg:

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = -\eta \sum_{p=1}^N \left(\sum_{i=1}^n w_i q_i^p - t^p \right) q_j^p,$$

Lernrate η frei wählbar



Durch Ersetzen der Aktivierung

$$y^p = \sum_{i=1}^n w_i q_i^p$$

des Ausgabeneurons bei angelegtem Trainingsbeispiel q^p :

Delta-Regel

$$\Delta w_j = \eta \sum_{p=1}^N (t^p - y^p) q_j^p.$$

DELTALERNEN(*Trainingsbeispiele*, η)

Initialisiere alle Gewichte w_j zufällig

Repeat

$$\Delta \mathbf{w} = \mathbf{0}$$

For all $(q^p, t^p) \in \text{Trainingsbeispiele}$

Berechne Netzausgabe $y^p = \mathbf{w}^p q^p$

$$\Delta \mathbf{w} = \Delta \mathbf{w} + \eta(t^p - y^p) \mathbf{q}^p$$

$$\mathbf{w} = \mathbf{w} + \Delta \mathbf{w}$$

Until w konvergiert

- Gewichtsänderungen erfolgen erst, nachdem alle Trainingsbeispiele angelegt wurden.

- Variante: direktes Ändern der Gewichte nach jedem Trainingsbeispiel

DELTALEARNENINKREMENTELL(*Trainingsbeispiele*, η)

Initialisiere alle Gewichte w_j zufällig

Repeat

For all $(q^p, t^p) \in \text{Trainingsbeispiele}$

Berechne Netzausgabe $y^p = w^p q^p$

$w = w + \eta(t^p - y^p)q^p$

Until w konvergiert

Inkrementelle Variante der Delta-Regel.

Vergleich mit dem Perzeptron

- Beim Perzeptron wird durch die Schwellwertentscheidung ein Klassifizierer für linear separable Klassen gelernt.
- Methode der kleinsten Quadrate und Delta-Regel erzeugen eine lineare Approximation an die Daten.
- Aus der gelernten linearen Abbildung kann durch Anwendung einer Schwellwertfunktion auch ein Klassifizierer erzeugt werden.

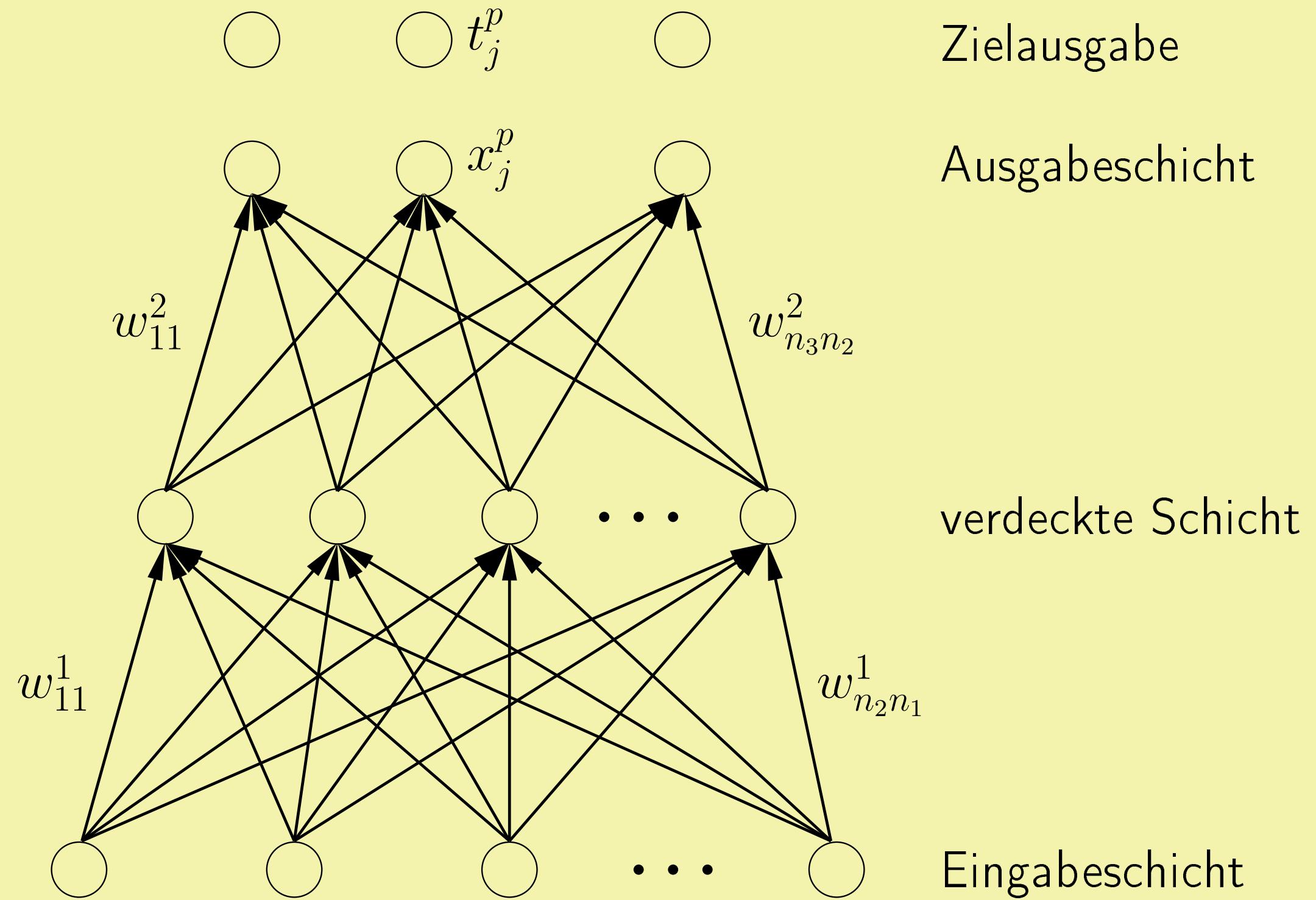
Methode der kleinsten Quadrate ist (u.a. wg. Rechenzeitgarantie) immer dann vorzuziehen, wenn das inkrementelle Lernen nicht benötigt wird.

Der Backpropagation-Algorithmus

- Erweiterung der inkrementellen Delta-Regel
- mit Sigmoid-Funktion
- mehr als zwei Schichten von Neuronen
- Bekannt durch den Artikel³ in der legendären PDP-Sammlung⁴.

³Rumelhart, D.E./Hinton, G.E./R.J., Williams Learning Internal Representations by Error Propagation. in Rumelhart/McClelland, 1986.

⁴Rumelhart, D./McClelland, J. Parallel Distributed Processing. Band 1, MIT Press, 1986.



Neuronenmodell:

$$x_j = f \left(\sum_{i=1}^n w_{ji} x_i \right) \quad (9.13)$$

mit der Sigmoidfunktion

$$f(x) = \frac{1}{1 + e^{-x}}.$$

Ähnlich wie bei der inkrementellen Deltaregel werden die Gewichte entsprechend dem negativen Gradienten der über die Ausgabeneuronen summierten quadratischen Fehlerfunktion

$$E_p(\mathbf{w}) = \frac{1}{2} \sum_{k \in \text{Ausgabe}} (t_k^p - x_k^p)^2$$

für das Trainingsmuster p geändert:

$$\Delta_p w_{ji} = -\eta \frac{\partial E_p}{\partial w_{ji}}.$$

Mehrfache Anwendung der Kettenregel (siehe⁵ oder⁶) liefert die
Backpropagation-Lernregel (verallgemeinerte Deltaregel)

$$\Delta_p w_{ji} = \eta \delta_j^p x_i^p,$$

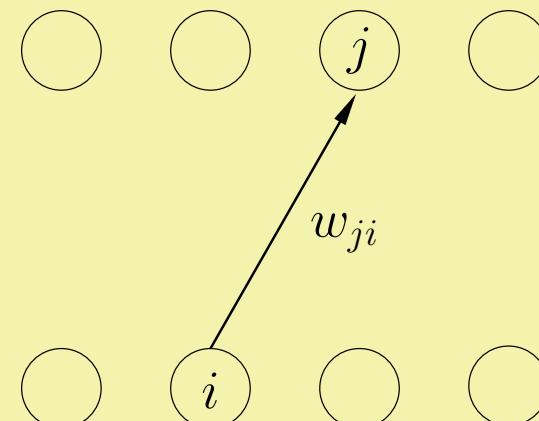
mit

$$\delta_j^p = \begin{cases} x_j^p(1 - x_j^p)(t_j^p - x_j^p) & \text{falls } j \text{ Ausgabeneuron ist} \\ x_j^p(1 - x_j^p) \sum_k \delta_k^p w_{kj} & \text{falls } j \text{ verdecktes Neuron ist,} \end{cases}$$

⁵Rumelhart, D.E./Hinton, G.E./R.J., Williams Learning Internal Representations by Error Propagation. in Rumelhart/McClelland, 1986.

⁶Zell, A. Simulation Neuronaler Netze. Addison Wesley, 1994.

Bezeichnungen der Neuronen und Gewichte für die Anwendung der Backpropagation-Regel.



BACKPROPAGATION(*Trainingsbeispiele*, η)

Initialisiere alle Gewichte w_j zufällig

Repeat

For all $(q^p, t^p) \in$ *Trainingsbeispiele*

1. **Anlegen des Anfragevektors** q^p an die Eingabeschicht
2. **Vorwärtspropagieren:**

Für alle Schichten ab der ersten verdeckten aufwärts

Für alle Neuronen der Schicht

Berechne Aktivierung $x_j = f\left(\sum_{i=1}^n w_{ji}x_i\right)$

3. **Berechnung des quadratischen Fehlers** $E_p(w)$

4. **Rückwärtspropagieren:**

Für alle Lagen von Gewichten ab der letzten abwärts

Für jedes Gewicht w_{ji}

$$w_{ji} = w_{ji} + \eta \delta_j^p x_i^p$$

Until w konvergiert oder Zeitschranke erreicht

- Es können nichtlineare Abbildungen gelernt werden.
- Ohne verdeckte Schicht nur lineare Abbildungen möglich.
- mehrlagige Netze mit linearer Aktivierungsfunktion können nur eine lineare Abbildungen lernen

variable Sigmoidfunktion

$$f(x) = \frac{1}{1 + e^{-(x-\Theta)}}.$$

mit Schwelle Θ verwendet.

Wie beim Perzeptron wird zu jeder Schicht ein Neuron hinzugefügt, dessen Aktivierung immer den Wert eins hat und das mit allen Neuronen der nächst höheren Schicht verbunden ist.

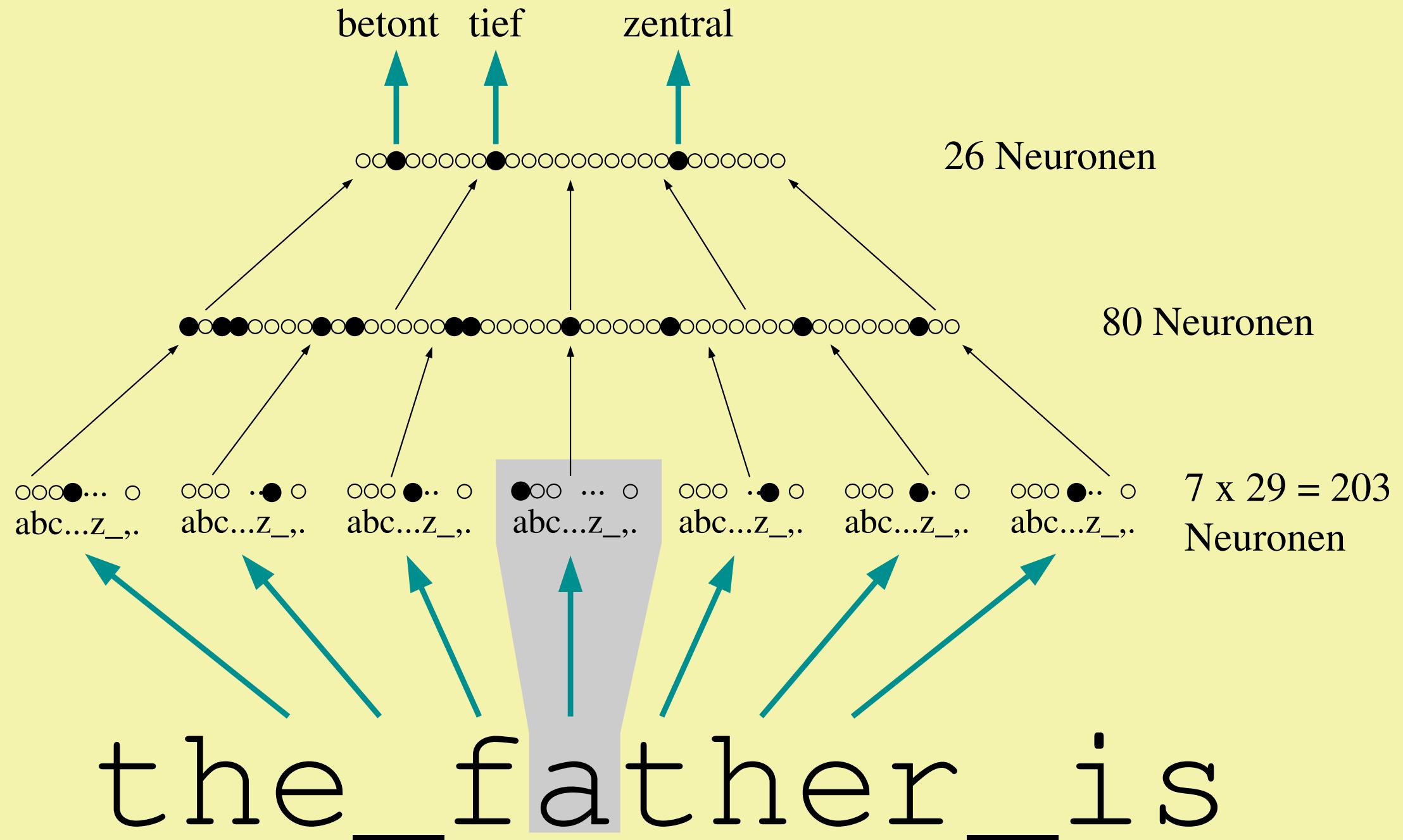
Die Gewichte von dieser Verbindungen werden ganz normal gelernt und repräsentieren die Schwellen Θ der Nachfolgeneuronen.

NETtalk: ein Netz lernt das Sprechen

Sejnowski und Rosenberg 1986⁷

System lernt, englischen Text laut und verständlich vorzulesen.

⁷ Sejnowski, T.J./Rosenberg, C.R. NETtalk: a parallel network that learns to read aloud. The John Hopkins University Electrical Engineering and Computer Science Technical Report, 1986 (JHU/EECS-86/01). – Technischer Bericht.



- Das Netz wurde trainiert mit 1000 Worten
- Für jeden Buchstaben wurde als Zielausgabe manuell dessen Betonung angegeben.
- Um die Betonungsattribute in reale Töne umzusetzen, wurde ein Teil des Sprachsynthesesystems DECTalk verwendet.
- Das Netz hat insgesamt $203 \times 80 + 80 \times 26 = 18320$ Gewichte.
- auf VAX 780 wurde mit 50 Zyklen über alle Worte trainiert.
- Bei 5 Zeichen pro Wort: $5 \cdot 50 \cdot 1000 = 250\,000$ Iterationen des Backpropagation-Algorithmus.
- 69 Stunden Rechenzeit.
- Menschliche Eigenschaften: Am Anfang undeutlich nur ganz einfache Worte.
- Später: Korrektheit von 95%.

JavaNNS

File Edit View Tools Pattern Window Help

Network: nettalk Training pattern set: nettalk Validation pattern set: nettalk

L>

Error graph

Cycles	Error
0	100
20	100
40	60
60	45
80	35
100	30
120	25
140	22
160	18
180	12
200	10

Learning cycles

Control Panel

Learning Pruning Patterns Subpatterns

Initializing Updating

Learning function: Backpropagation

Parameters

η : 0.2 d_{max} : 0.1

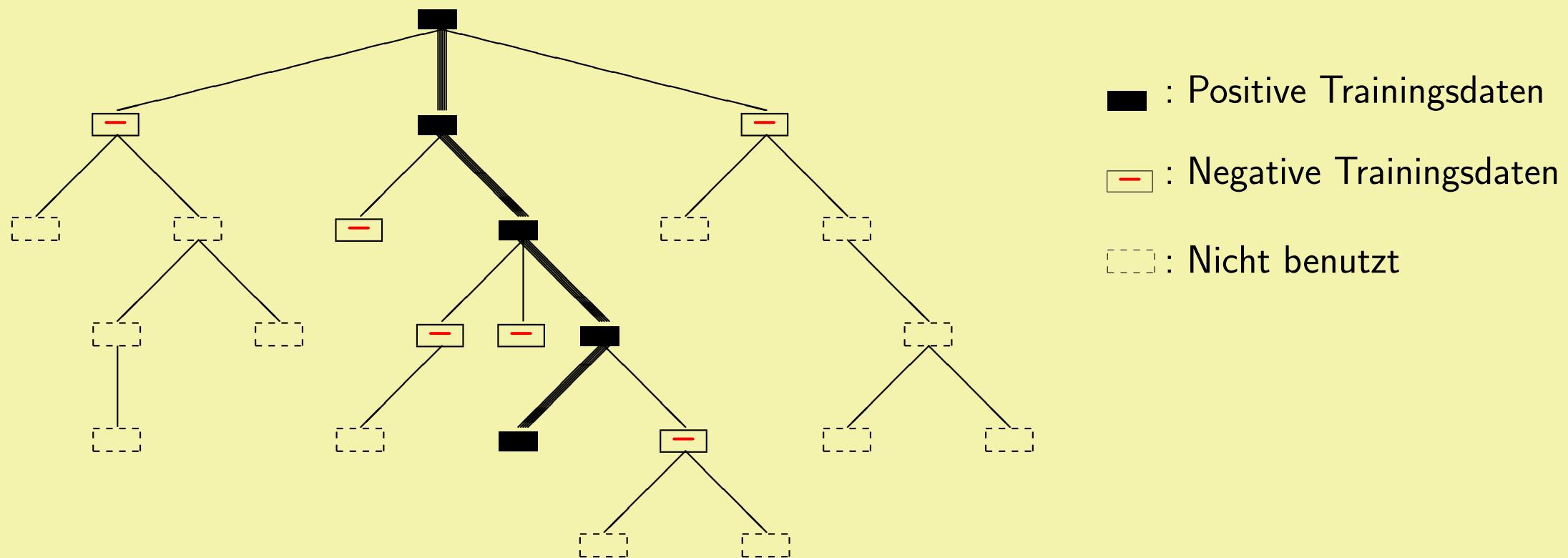
Cycles: 200 Steps: 1 Shuffle

Init **Learn Current** **Learn All**

Lernen von Heuristiken für Theorembeweiser

- Semantic Web benötigt automatische Theorembeweiser, um Suchanfragen zu beantworten
- große Zahl möglicher Inferenzschritte beim Beweisen
- kombinatorische Explosion des Suchraums
- heuristische Suche: z.B. A[★]-Algorithmus
- **heuristische Bewertungsfunktion ???**

Suchbaum eines Beweisers



- verschiedenen Alternativen für den nächsten Schritt bewerten
- Alternative mit der besten Bewertung auswählen.
- Resolution: Bewertung der verfügbaren Klauseln basierend auf Attributen wie: Zahl der Literale, Zahl der positiven Literale, Komplexität der Terme, etc.
- Positive und negative Trainingsdaten
- Auf diesen Daten wird ein Backpropagation-Netzwerk trainiert
- Mit dem gelernten Netz ist der Beweiser viel schneller (siehe^{8,9})

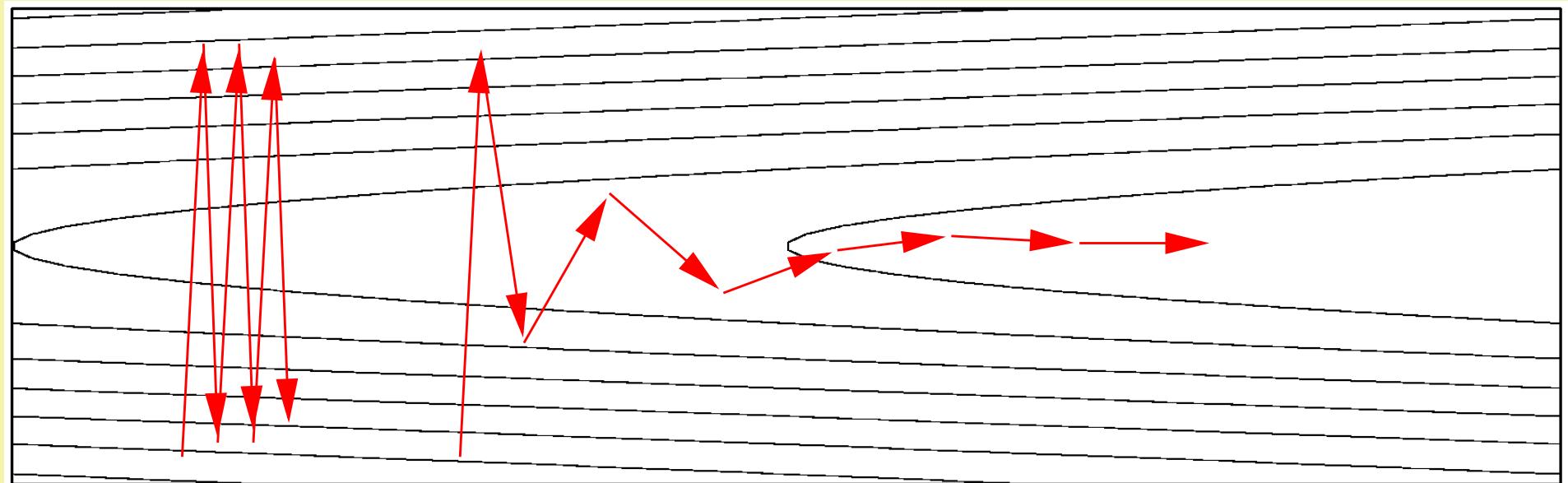
⁸Ertel, W./Schumann, J./Suttner, Ch. Learning Heuristics for a Theorem Prover using Back Propagation. In Retti, J./Leidlmaier, K. (Hrsg.) 5. Österreichische Artificial-Intelligence-Tagung. Berlin, Heidelberg: Informatik-Fachberichte 208, Springer-Verlag, 1989.

⁹Suttner, Ch./Ertel, W. Automatic Acquisition of Search Guiding Heuristics. In 10th Int. Conf. on Automated Deduction. Springer-Verlag, LNAI 449, 1990.

Probleme und Verbesserungen

- lokale Minima der
- Konvergenz von Backpropagation oft sehr langsam
- Verwendung eines Impuls-Terms (engl. momentum) bei der Gewichtsänderung.
Die Lernregel ändert sich dann zu

$$\Delta_p w_{ji}(t) = \eta \delta_j^p x_i^p + \gamma \Delta_p w_{ji}(t - 1)$$



- Minimierung der linearen Fehlerfunktion statt der quadratischen

- quadratische Näherung der Fehlerfunktion (Fahlmann): Quickprop
- Adaptive Schrittweitensteuerung (Riedmiller): RProp

Support-Vektor-Maschinen

Vorteile von linearen neuronalen Netzen:

- schnelles Lernen
- Konvergenzgarantie
- geringe Gefahr für Overfitting

Vorteile von nichtlinearen Netzen (z.B. Backpropagation):

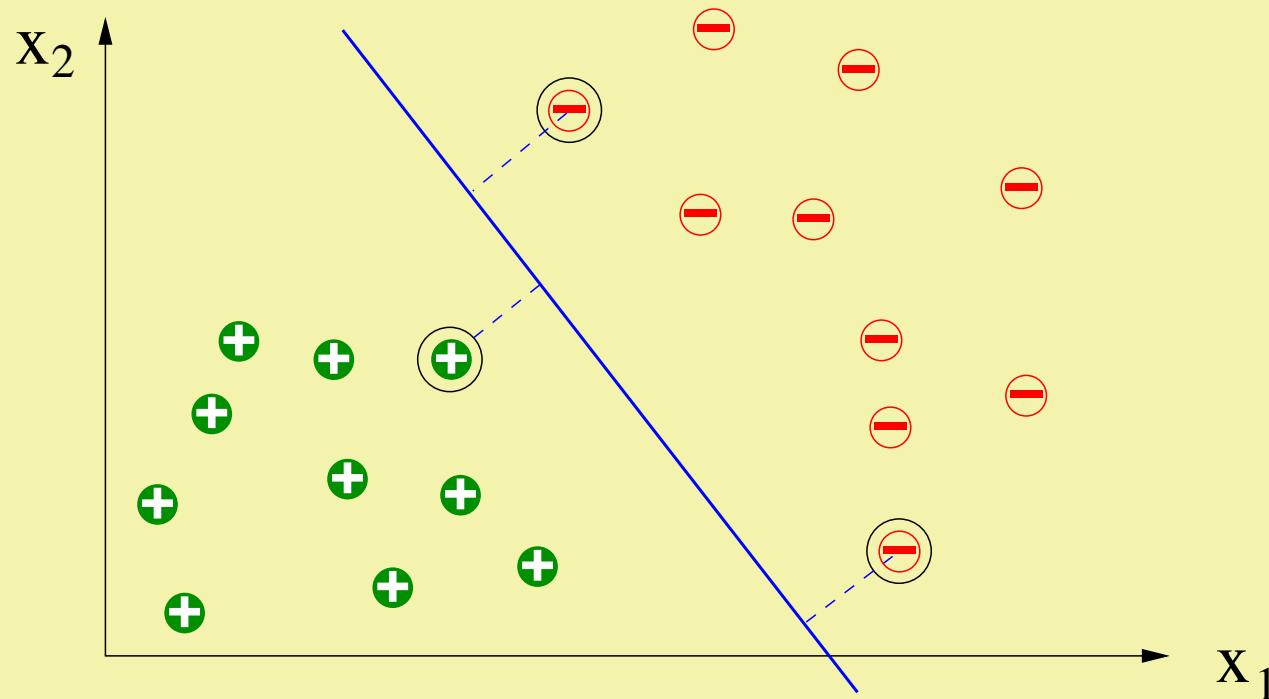
- komplexe Funktionen können gelernt werden

Nachteile von nichtlinearen Netzen (z.B. Backpropagation):

- lokale Minima, Konvergenzprobleme und Overfitting

Lösung: Support-Vektor-Maschinen (SVM)

- nichtlineare Transformation der Daten mit der Eigenschaft, dass die transformierten Daten linear separabel sind. Die Transformation wird als **Kernel** bezeichnet.
- Im transformierten Raum werden die Support-Vektoren bestimmt.



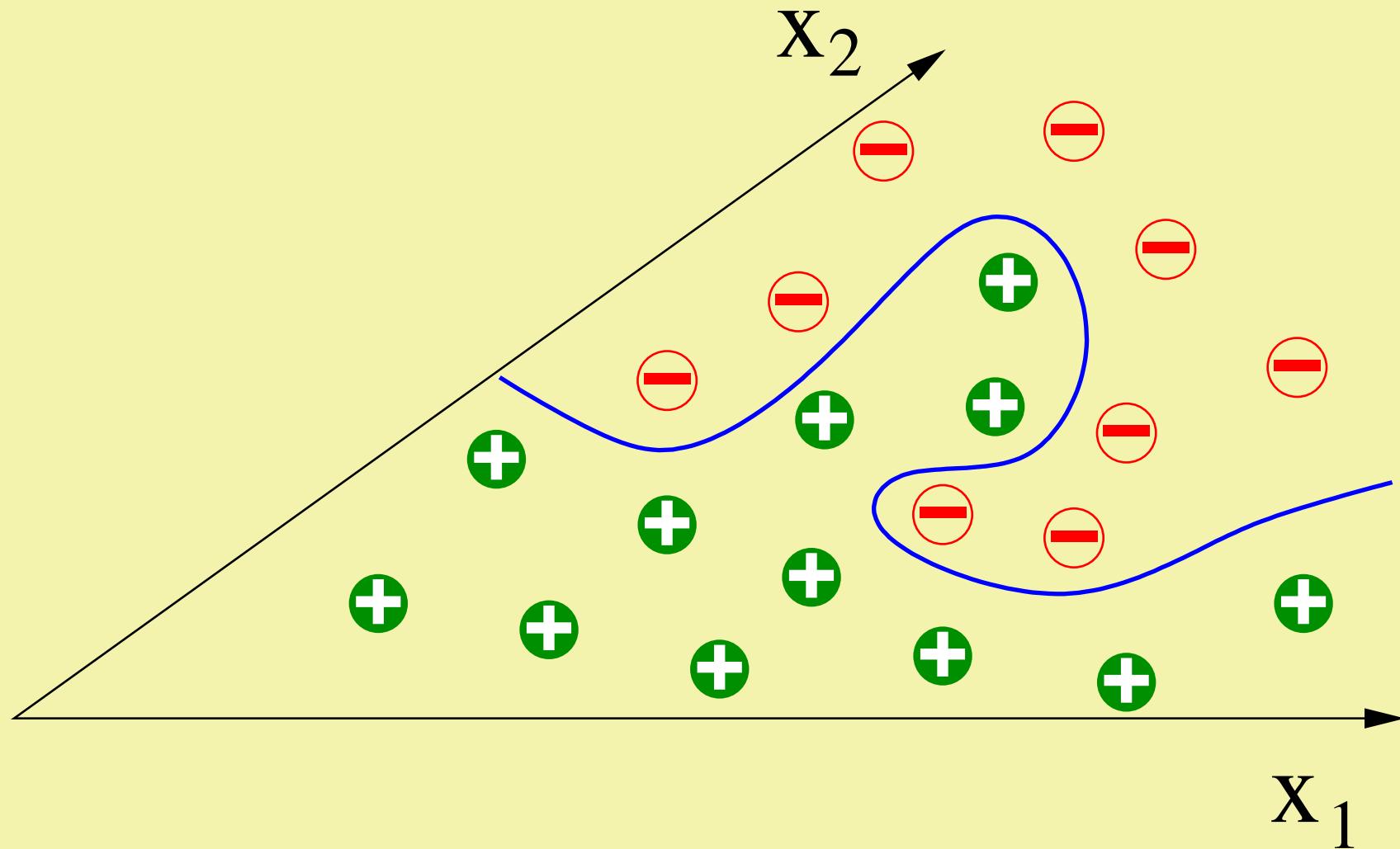
Lineare Trennung der Klassen:

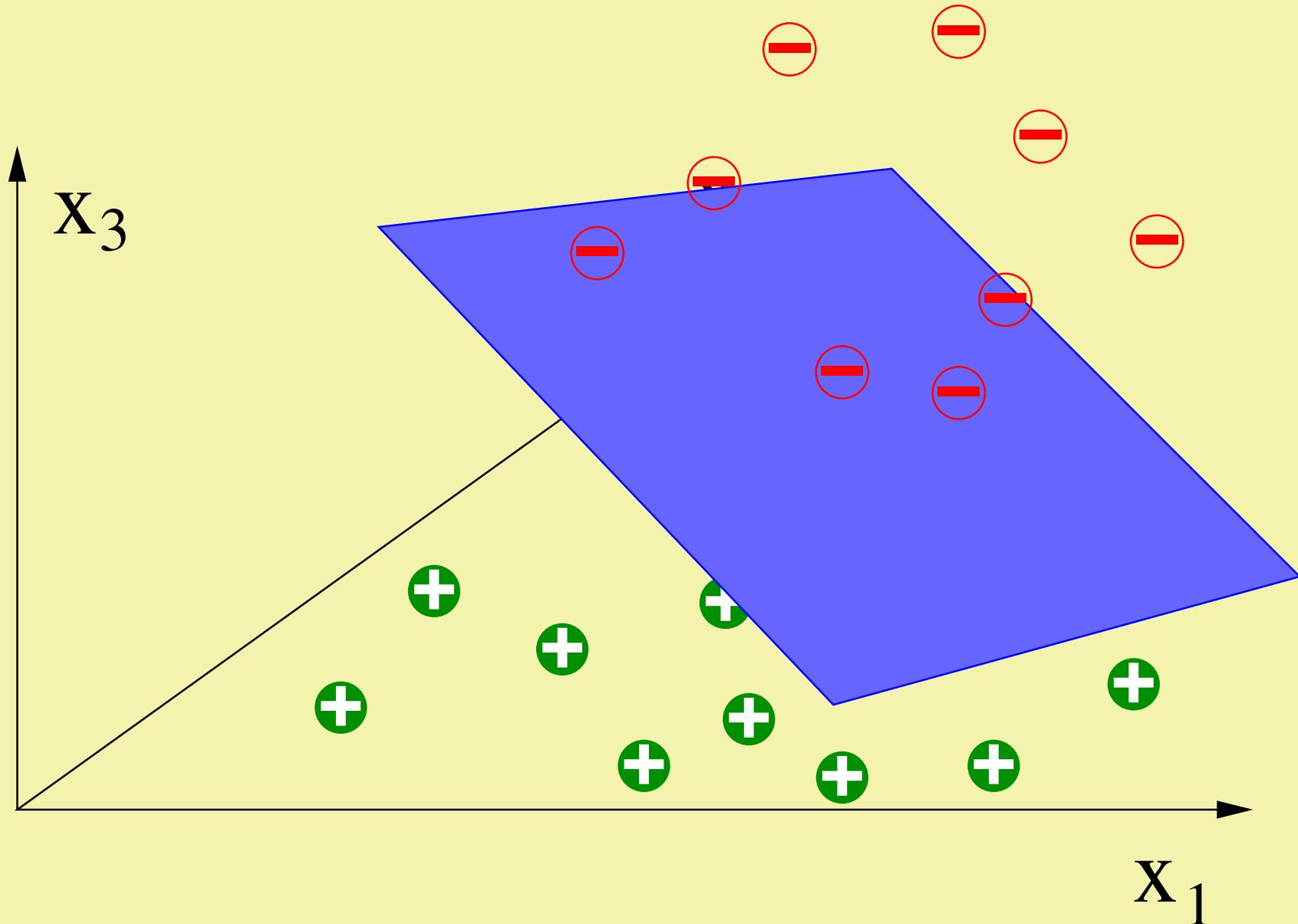
Es ist immer möglich, durch Transformation des Vektorraumes die Klassen linear separabel zu machen, sofern die Daten keine Widersprüche enthalten.¹⁰

z.B. mit der Transformation der Daten durch Einführung einer neuen Dimension x_{n+1} :

$$x_{n+1} = \begin{cases} 1 & \text{falls } x \in \text{Klasse 1} \\ 0 & \text{falls } x \in \text{Klasse 0.} \end{cases}$$

¹⁰ Ein Datenpunkt ist widersprüchlich, wenn er zu beiden Klassen gehört.





So einfach geht es aber nicht. Warum?

- SVMs sind auch auf Regressionsprobleme anwendbar.

Literatur:

- Schölkopf, S./Smola, A. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, 2002
- Alpaydin, E. Introduction to Machine Learning. MIT Press, 2004
- Burges, C. J. A Tutorial on Support Vector Machines for Pattern Recognition. Data Min. Knowl. Discov. 2 1998, Nr. 2

Anwendungen Neuronaler Netze

- In allen Industriebereichen
- Mustererkennung in allen Ausprägungen
- Analyse von Fotos zur Erkennung von Personen oder Gesichtern
- Erkennen von Fischschwärmern auf Sonarechos
- Erkennen und Klassifizieren von militärischen Fahrzeugen auf Radarscans
- Erkennen von gesprochener Sprache und handschriftlichen Texten
- Steuern von einfachen Robotern
- heuristische Steuerung der Suche in Backgammon- oder Schachcomputern
- Einsatz der Netze zum Lernen durch Verstärkung (Kapitel 10)
- Prognose von Aktienkursen
- Beurteilung der Kreditwürdigkeit von Bankkunden
- ...

Zusammenfassung und Ausblick

- Perzeptron, Delta-Regel und Backpropagation
- verwandt zu Scores, Naive-Bayes und Methode der kleinsten Quadrate
- Hopfield-Netze sehr anscheulich, aber nur schwer in der Praxis handhabbar
- Für die Praxis wichtig: Assoziativspeichermodelle: Kohonen, Plam

Für alle NN-Modelle gilt:

- Informationen werden verteilt über viele Gewichte gespeichert (holografisch).
- Absterben einzelner Neuronen hat kaum Auswirkungen auf die Funktion eines Gehirns.
- Netzwerk ist robust gegenüber kleinen Störungen.
- Wiedererkennen von fehlerhaften Mustern

Nachteile:

- Es ist schwierig, Informationen zu lokalisieren.
- Es ist praktisch unmöglich, in einem trainierten Netz die vielen Gewichte zu analysieren und zu verstehen (⇒ Entscheidungsbaum)
- das „Großmutter-Neuron“ ist sehr schwer zu finden
- Probleme bei der Verbindung von NNs mit symbolverarbeitenden Systemen.

Nicht behandelt:

- selbstorganisierende Karten (Kohonen)
- inkrementelles Lernen
- sequentielle Netze zum Lernen von zeitabhängigen Prozessen
- Lernen ohne Lehrer (siehe Kapitel 10)

Literatur:

- Zell, A. Simulation Neuronaler Netze. Addison Wesley, 1994,
- Ritter, H./Martinez, T./Schulten, K. Neuronale Netze. Addison Wesley, 1991,
- Rojas, R. Theorie der neuronalen Netze. Springer, 1993.

Sammlung wichtiger Originalarbeiten in

- Anderson, J./Rosenfeld, E. Neurocomputing: Foundations of Research. Cambridge, MA: MIT Press, 1988,
- Anderson, J./Pellionisz, A./Rosenfeld, E. Neurocomputing (vol. 2): directions for research. Cambridge, MA, USA: MIT Press, 1990

Zeitschriften

- Neural Networks
- Neurocomputing

Konferenz: NIPS (Neural Information Processing Systems)

Kapitel 10

Lernen durch Verstärkung (Reinforcement Learning)

Einführung

- Robotik
- Aufgaben sind oft sehr komplex
- nicht programmierbar

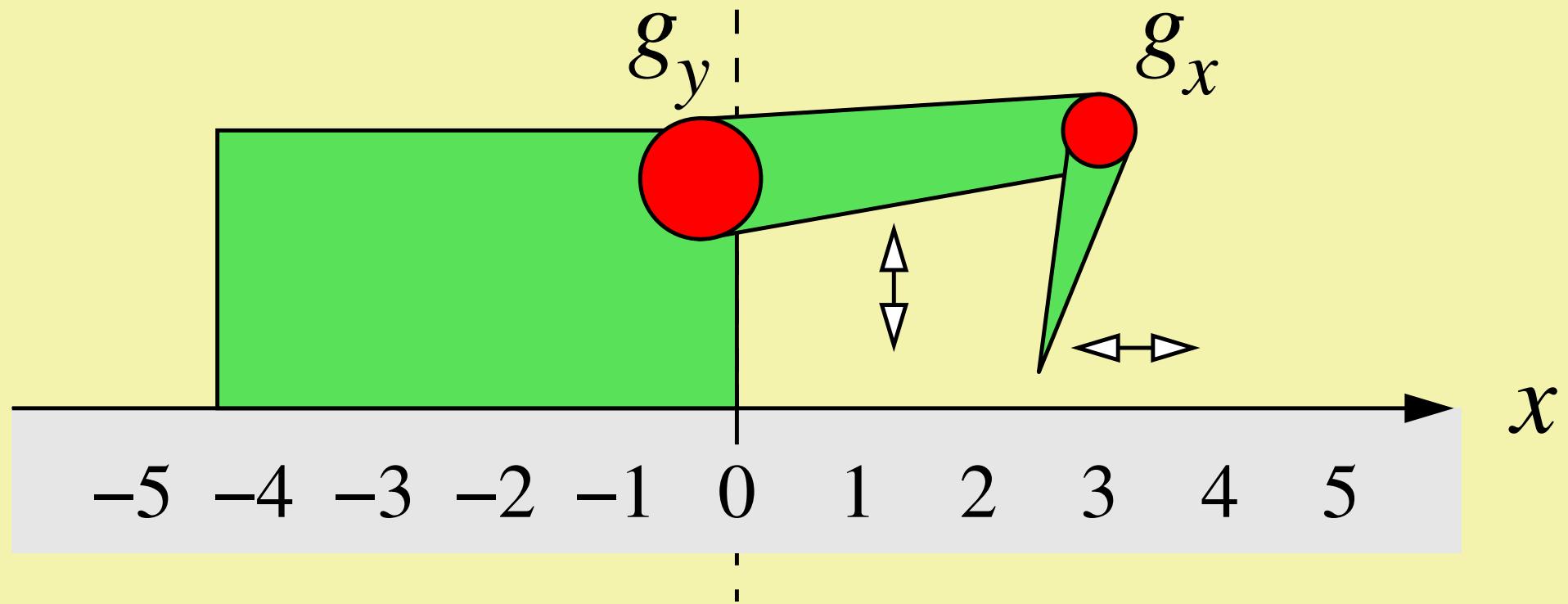
Aufgabenstellung

- durch Versuch und Irrtum herausfinden, welche Aktionen **gut** sind
- Menschen lernen so, z.B. Laufen:
- Belohnung durch Vorwärtskommen
- Bestrafung durch Stürze

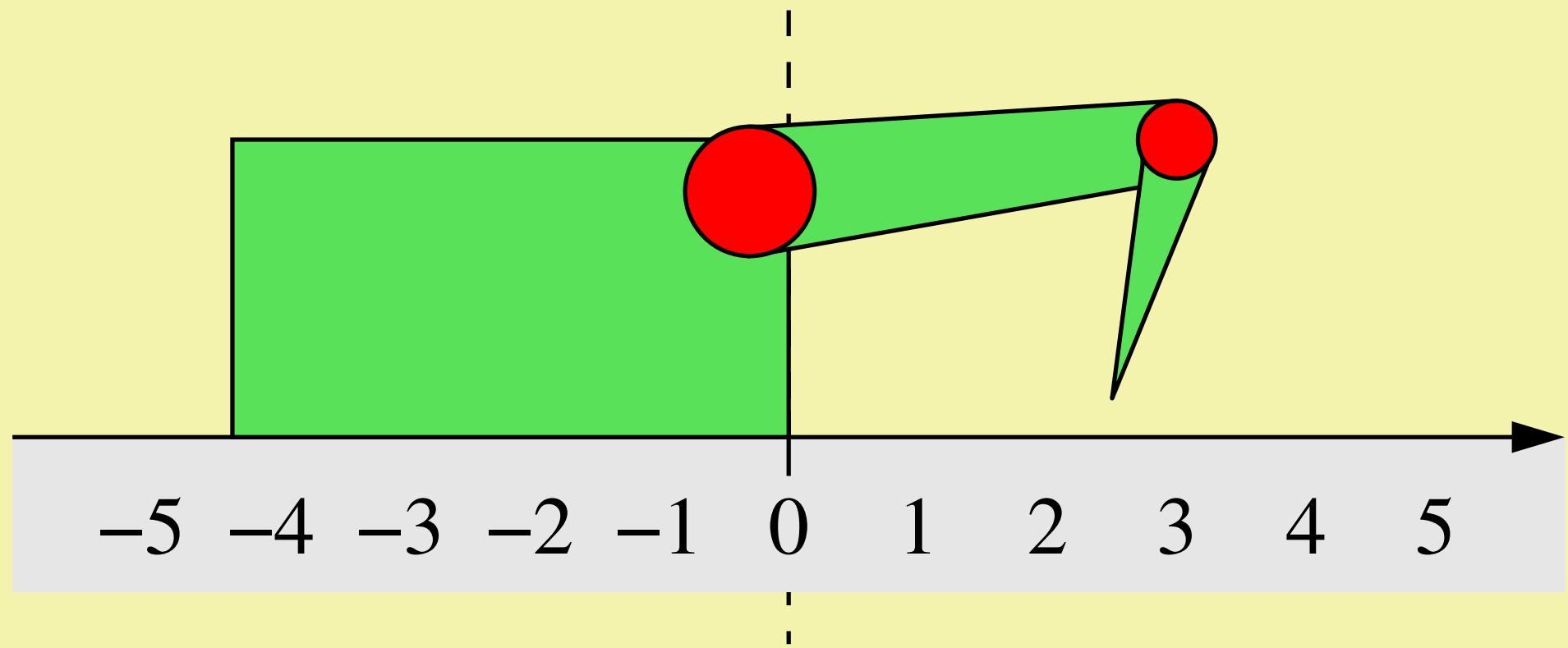


„Vielleicht sollte ich beim nächsten mal den Schwung etwas früher einleiten oder
die Kurve langsamer ansetzen.“

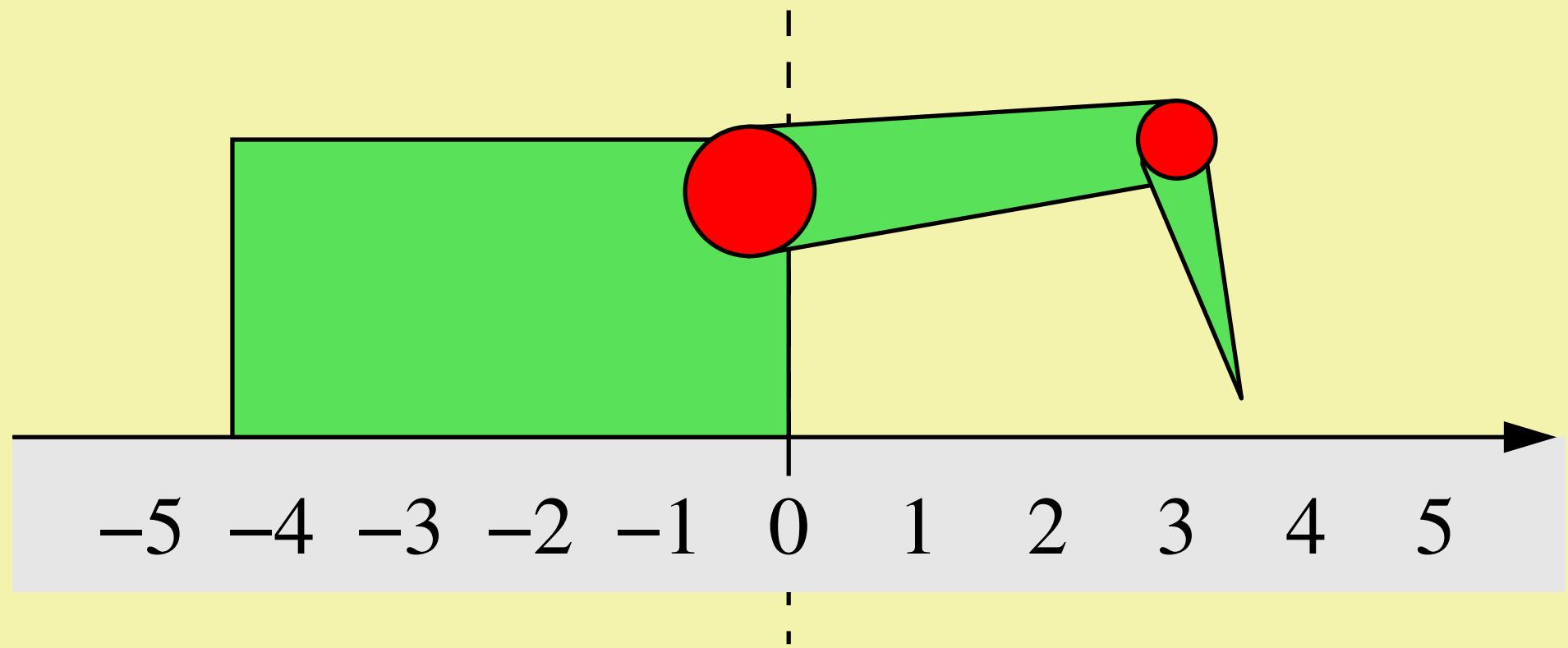
Der Krabbler



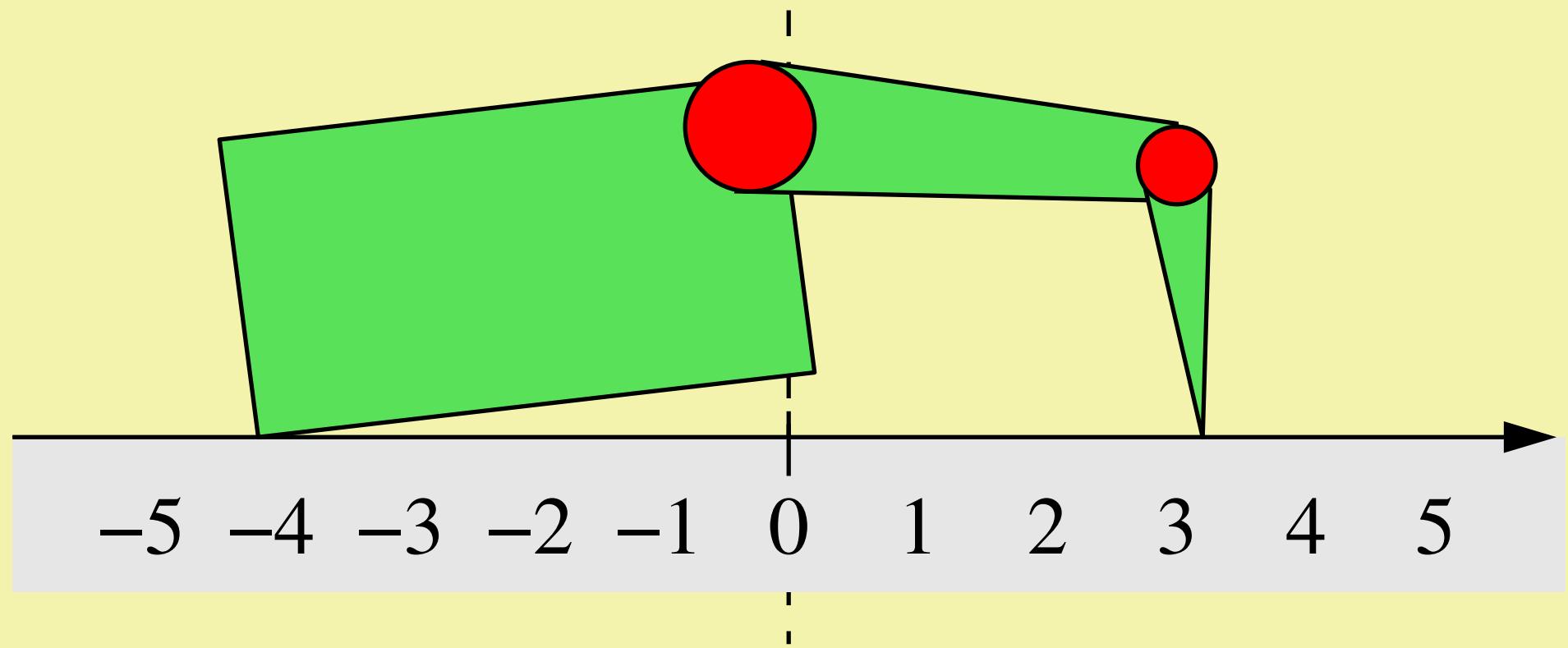
Der Krabbler



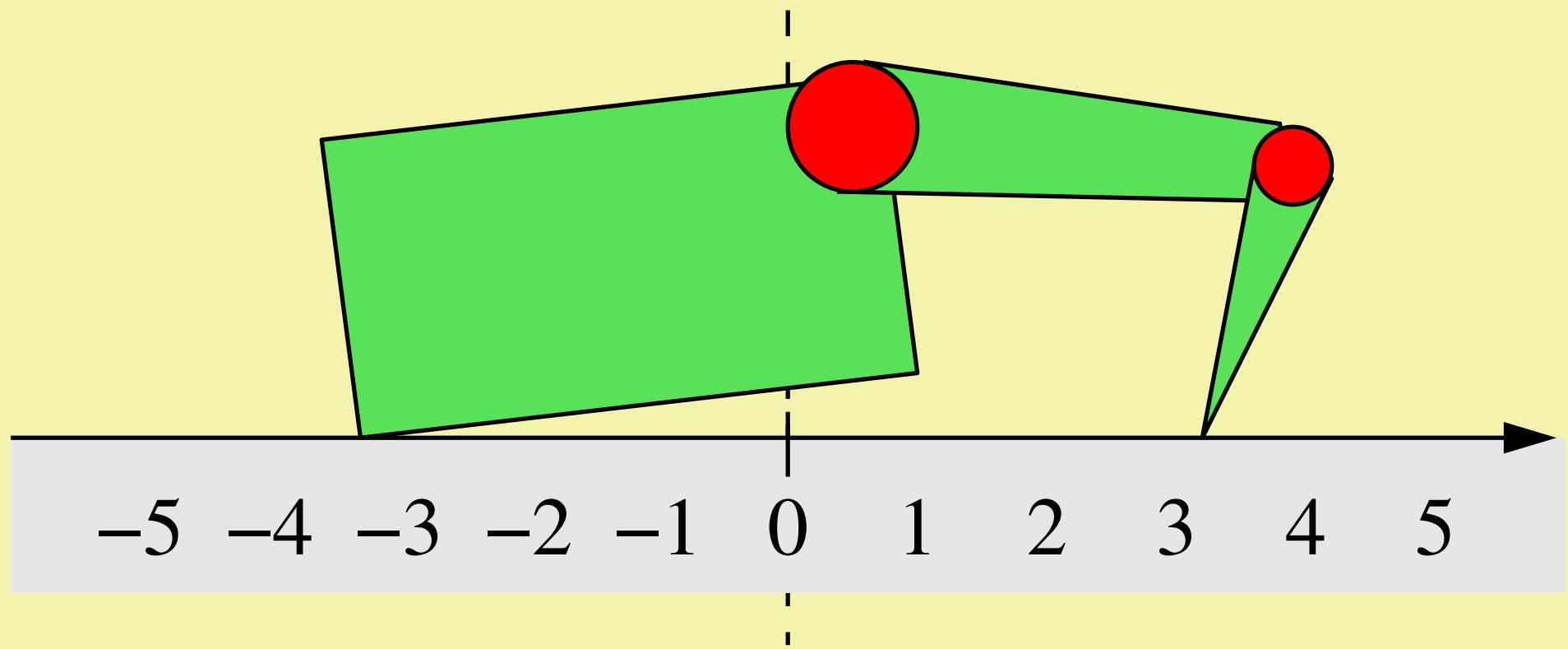
Der Krabbler



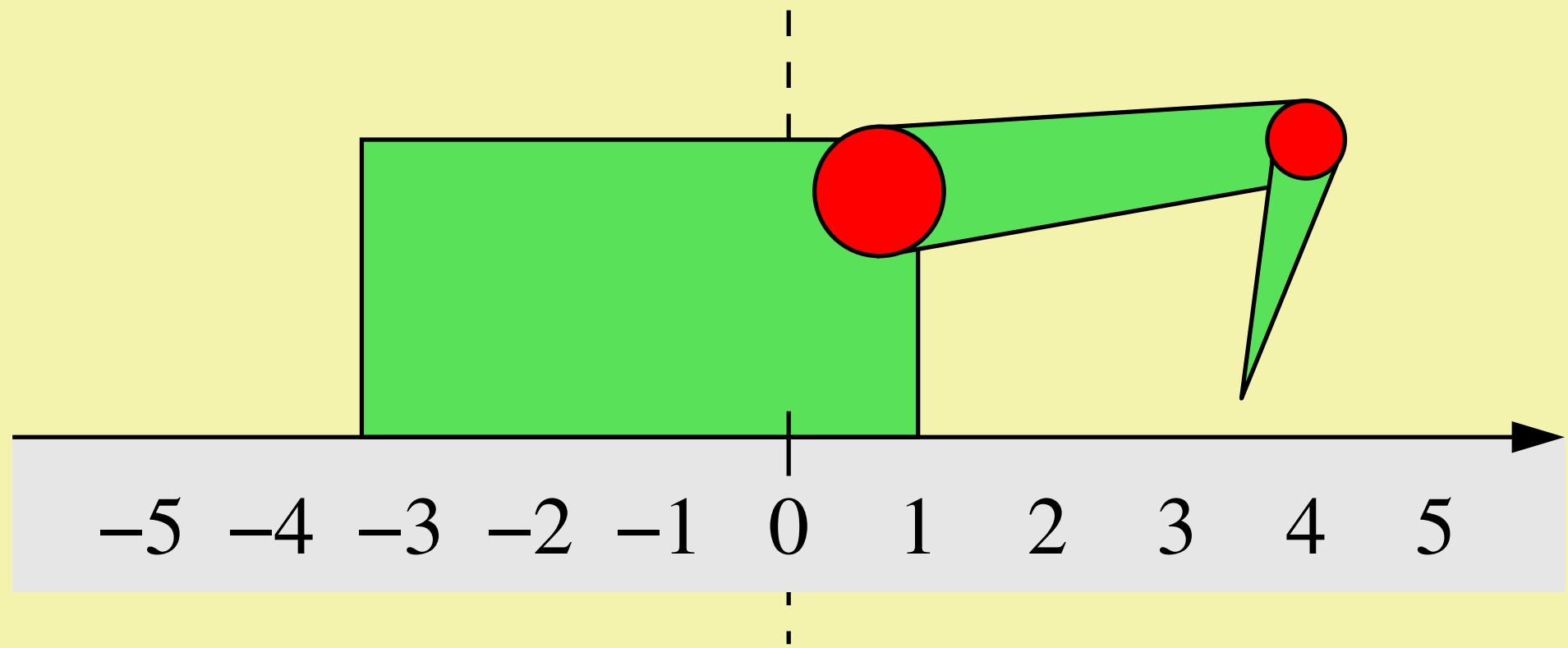
Der Krabbler



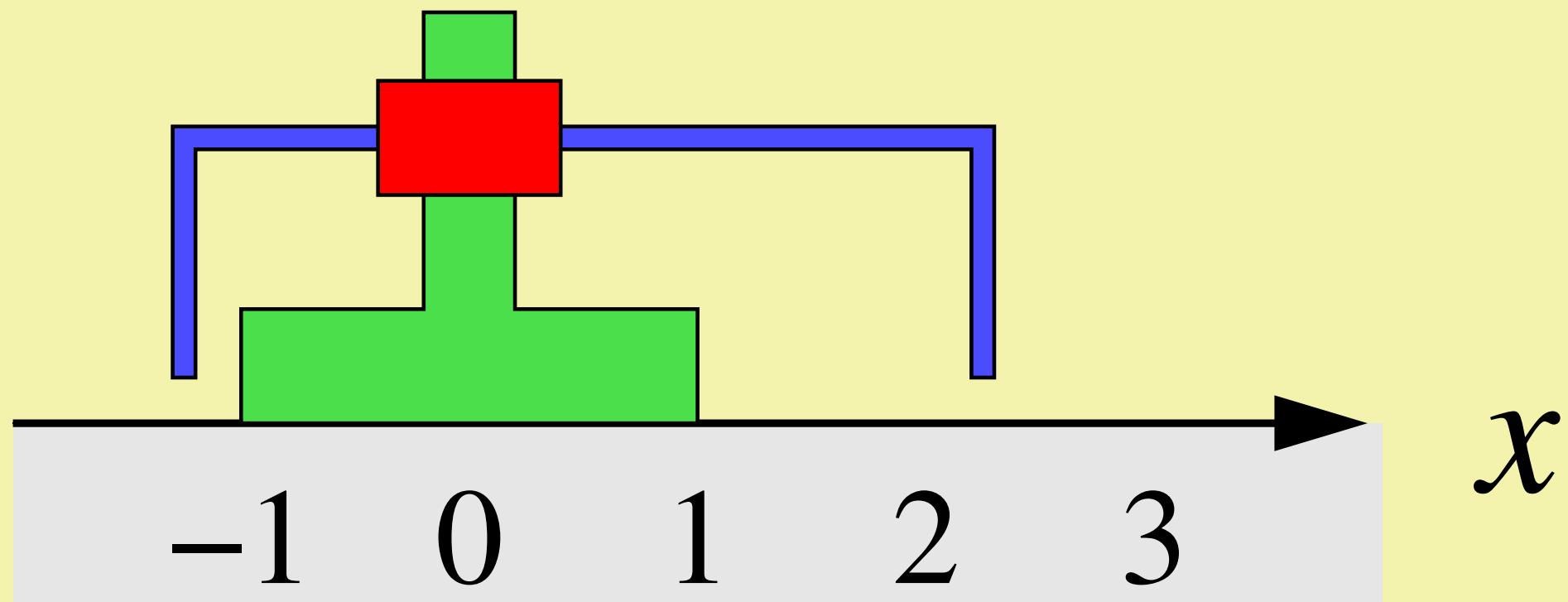
Der Krabbler



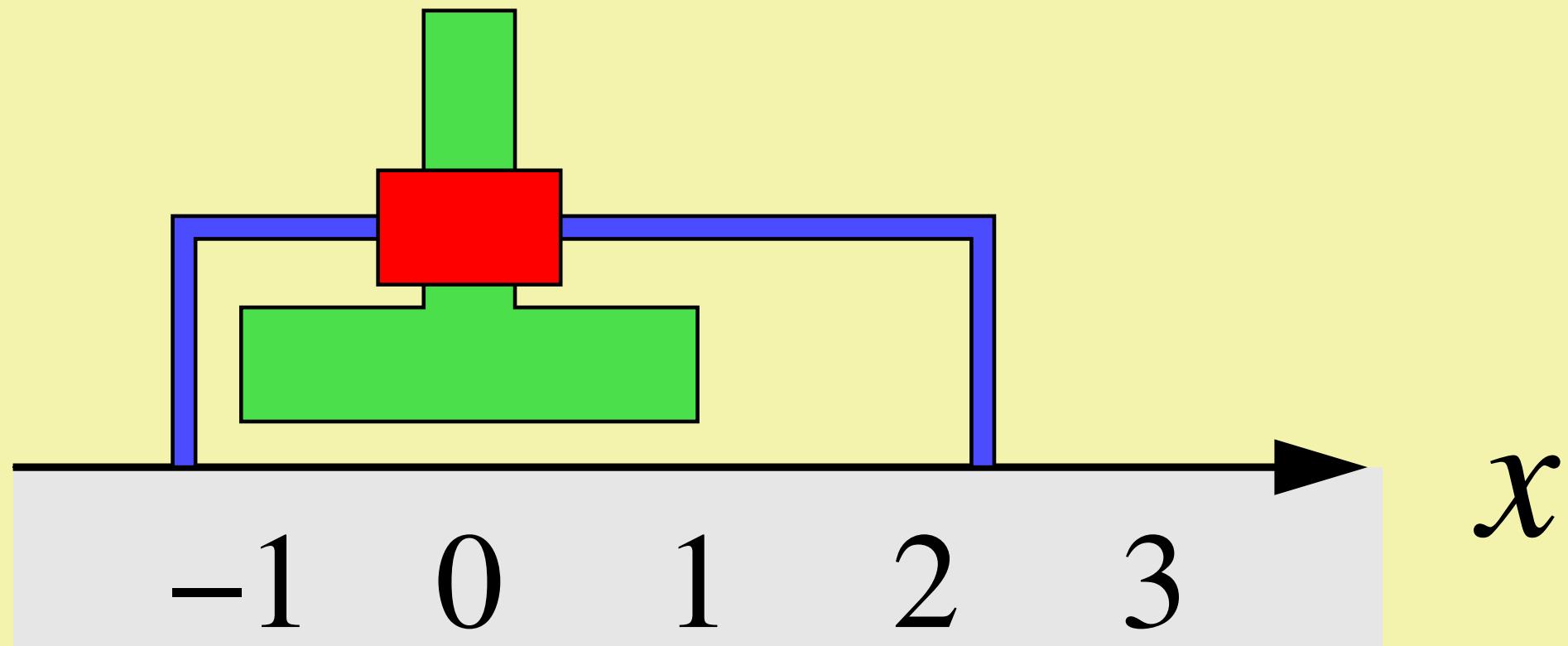
Der Krabbler



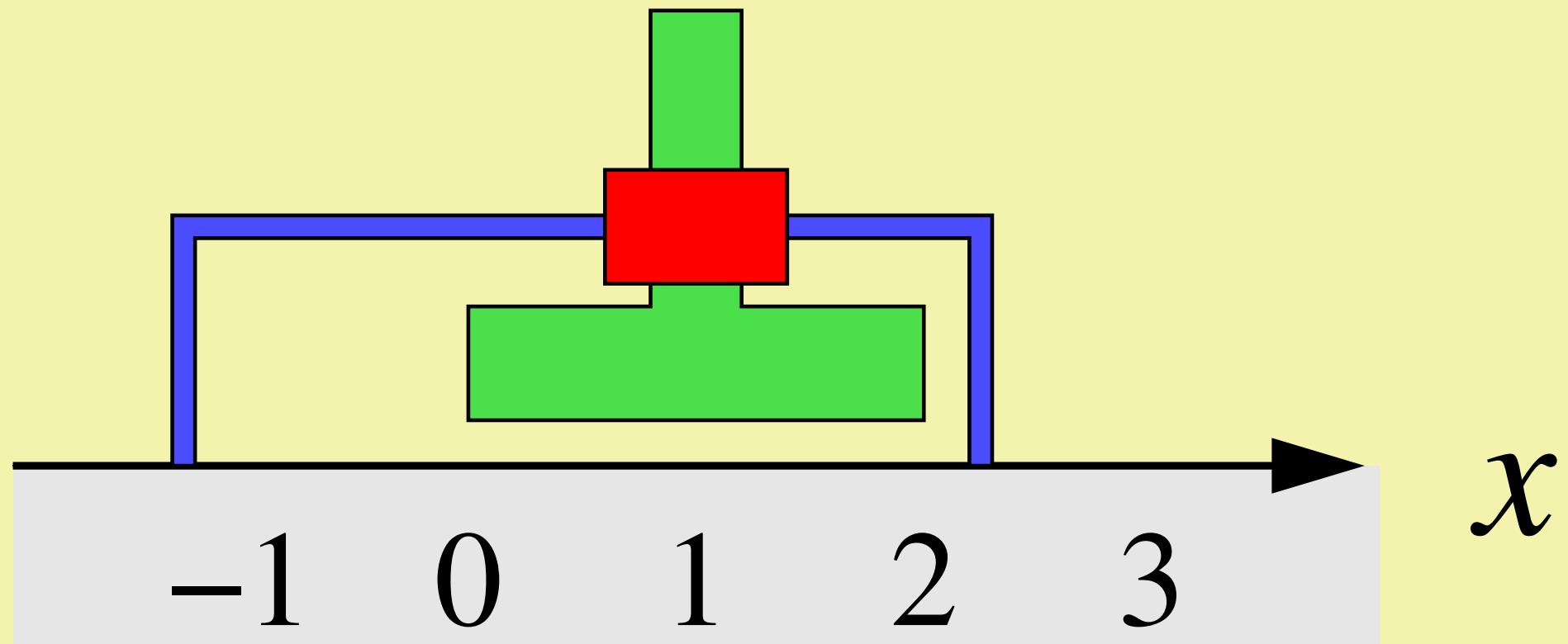
Der Laufroboter



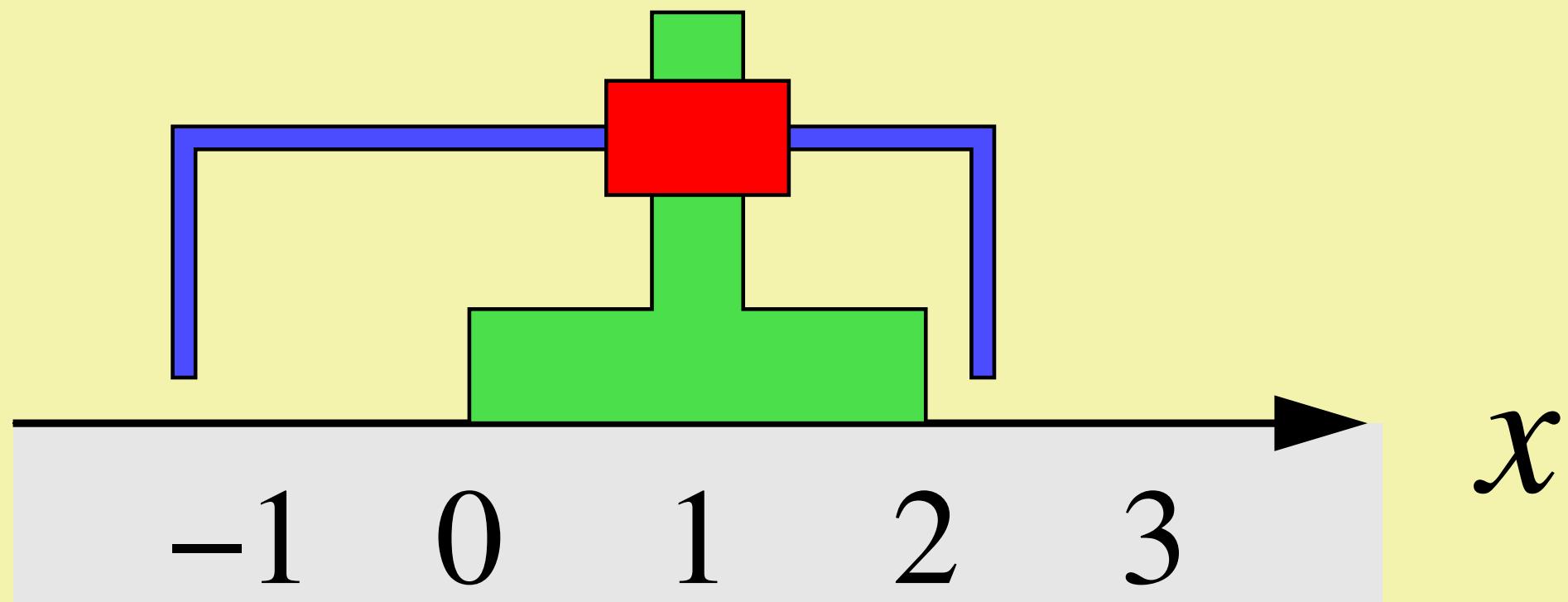
Der Laufroboter



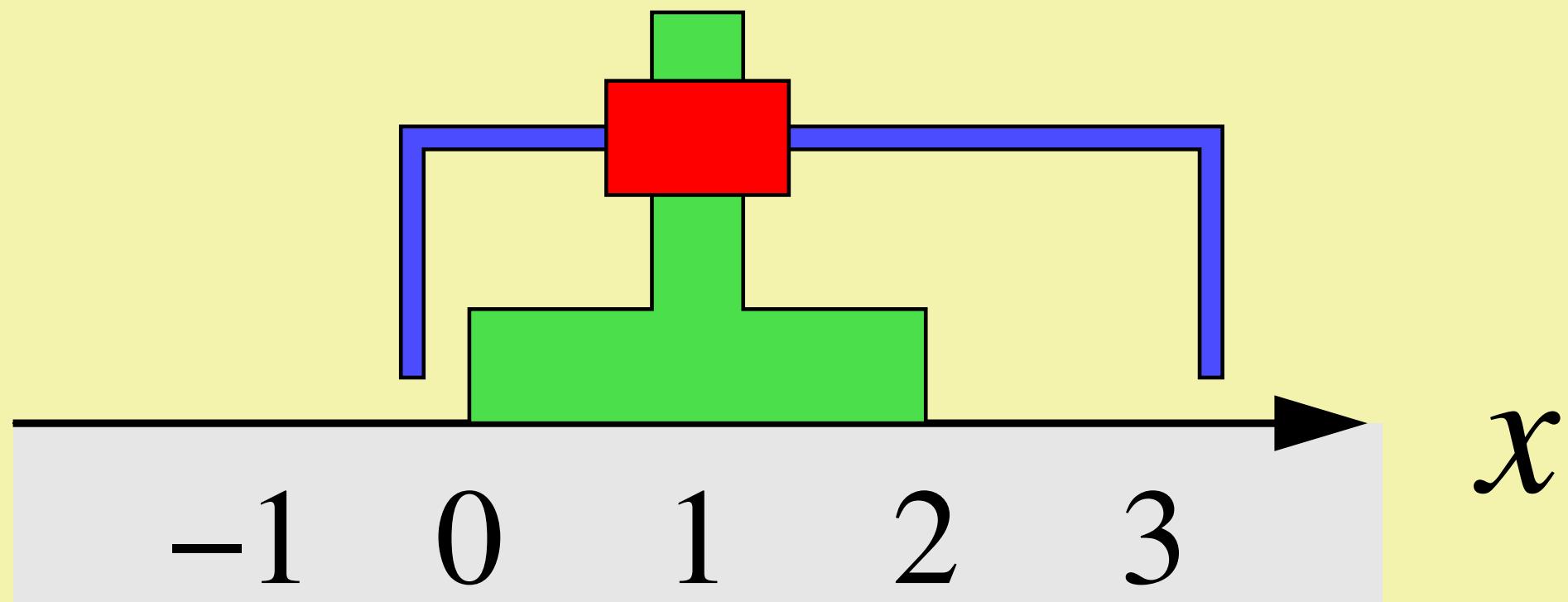
Der Laufroboter



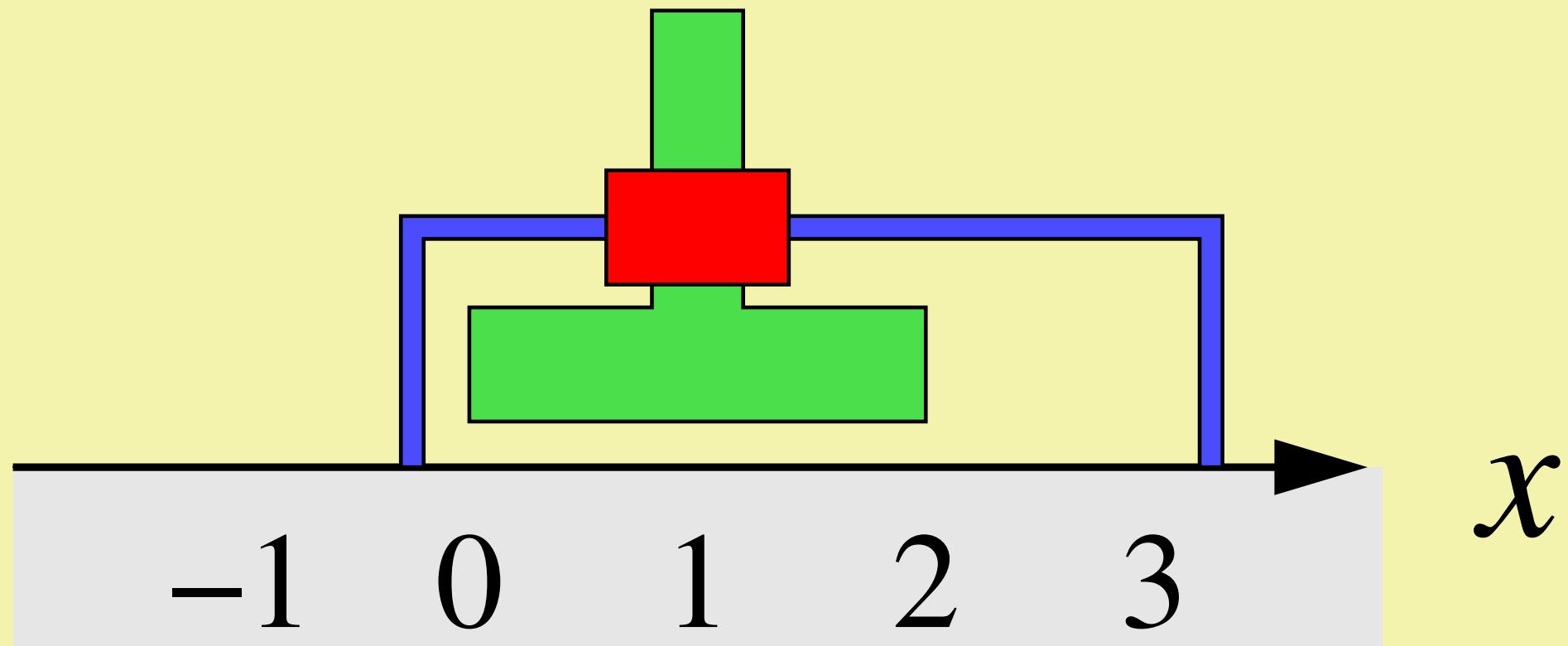
Der Laufroboter



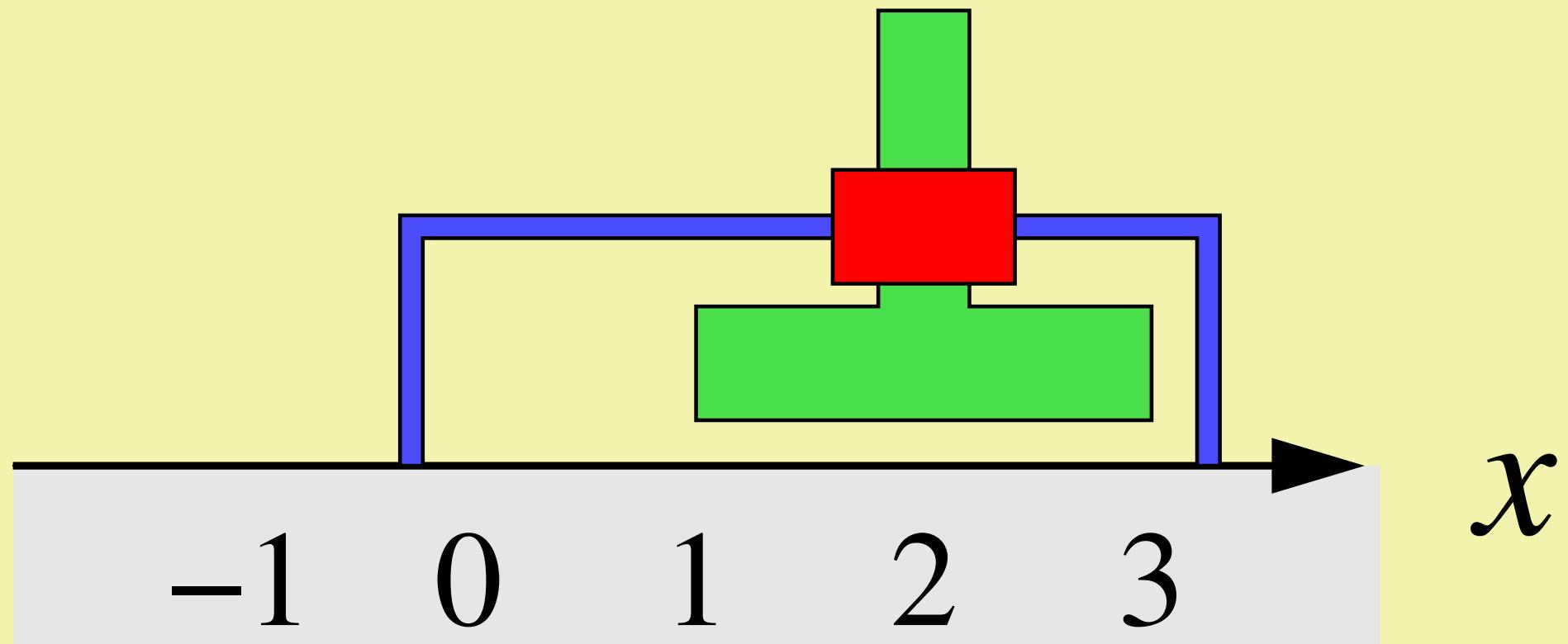
Der Laufroboter



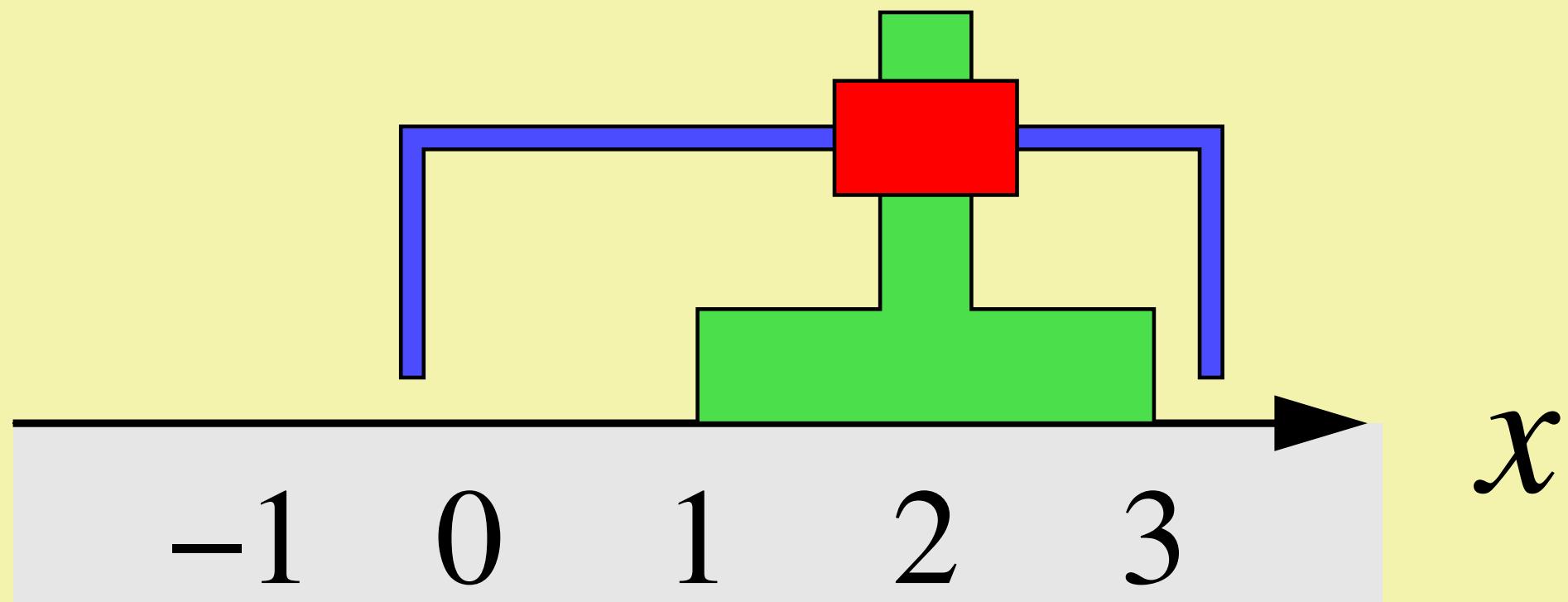
Der Laufroboter



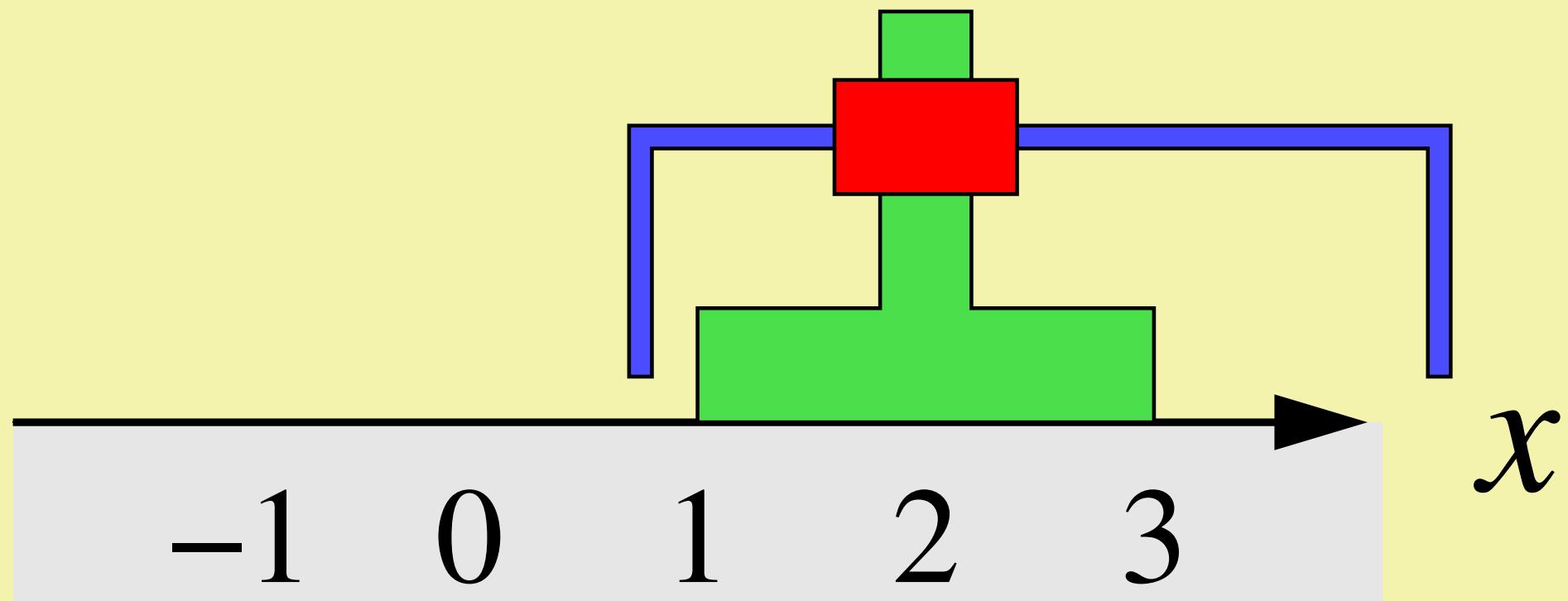
Der Laufroboter



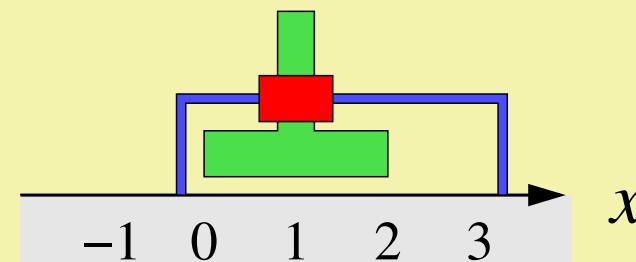
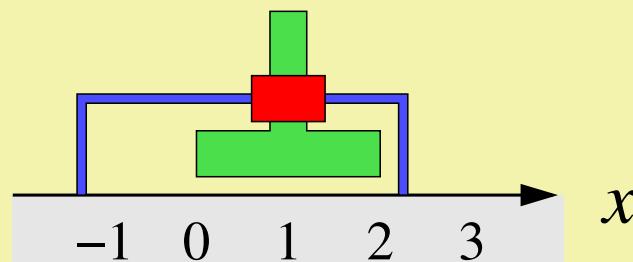
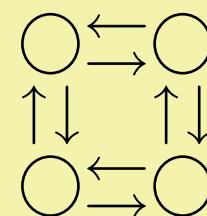
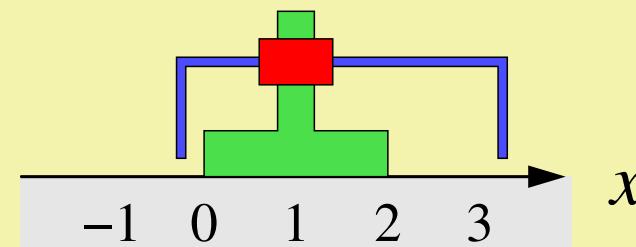
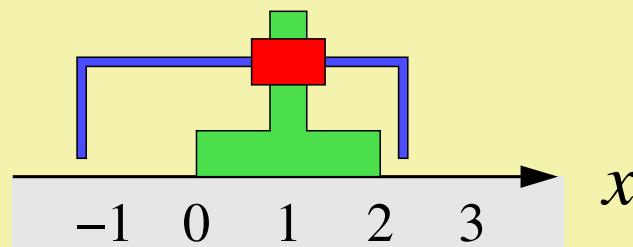
Der Laufroboter



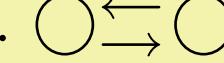
Der Laufroboter

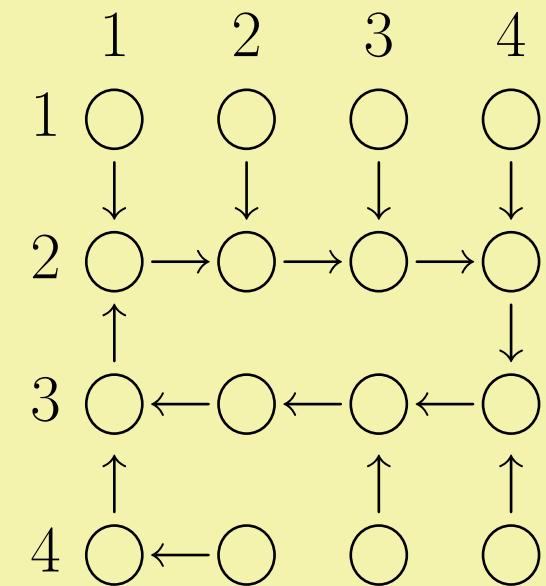
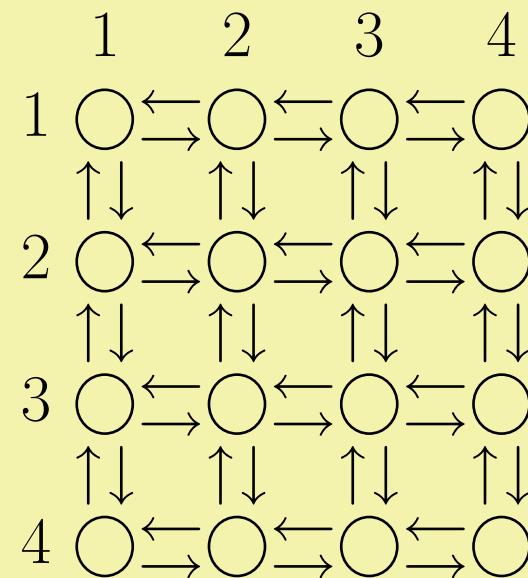


Der Zustandsraum



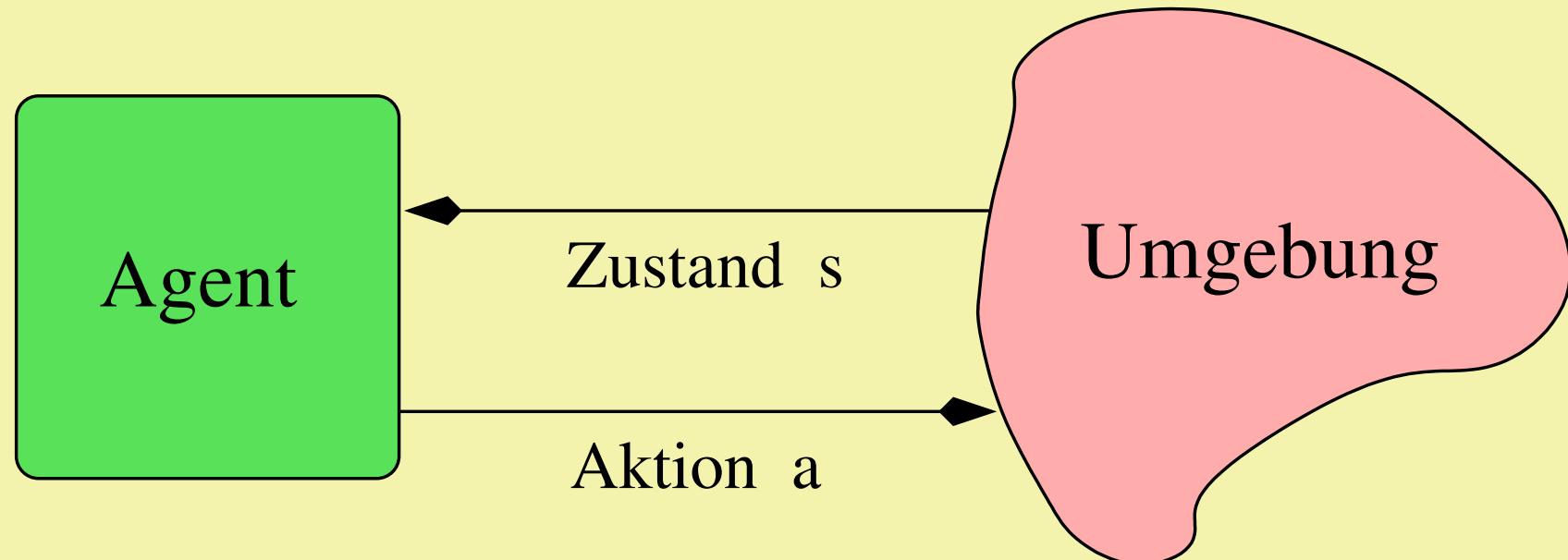
Der Zustandsraum

li. re.
 ob. 
 unt. 

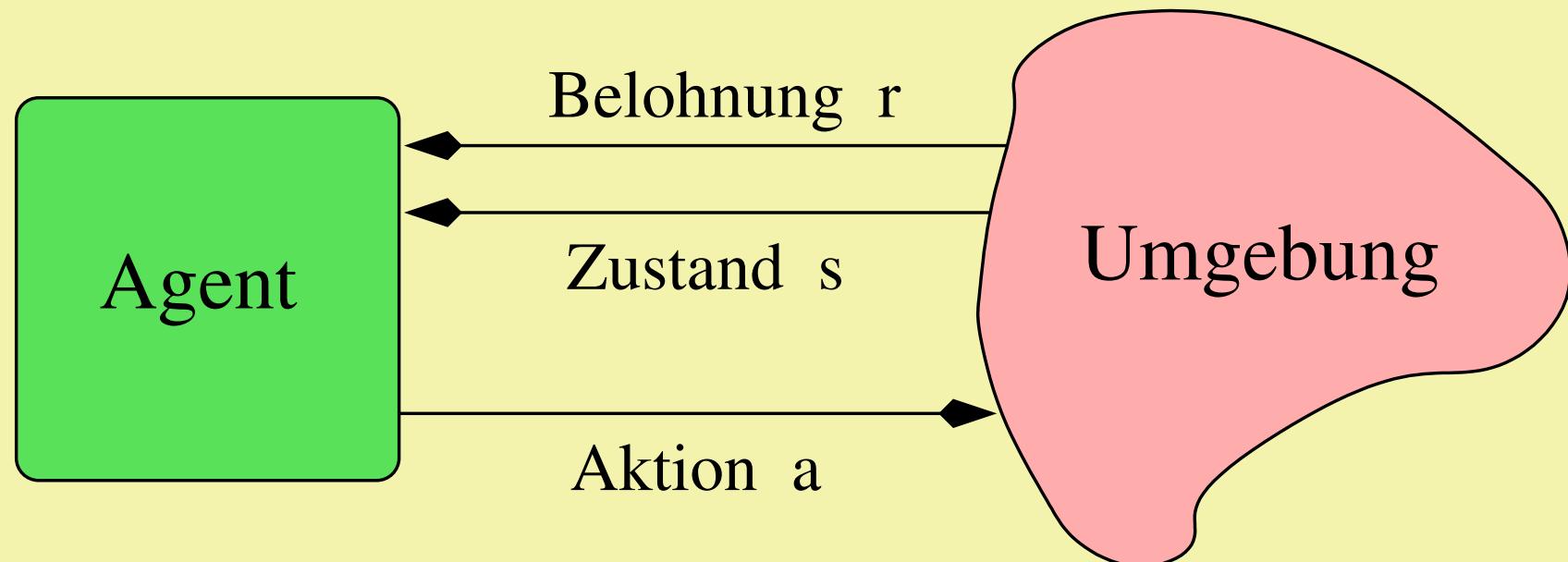


Zustandsraum: 2×2 (links), 4×4 (Mitte), optimale Strategie (rechts).

Der Agent



Der Agent



Die Aufgabenstellung

Zustand $s_t \in \mathcal{S}$:

$$s_t \xrightarrow{a_t} s_{t+1}$$

Übergangsfunktion δ :

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

Direkte Belohnung

$$r_t = r(s_t, a_t)$$

$r_t > 0$: positive Verstärkung

$r_t = 0$: kein Feedback oft ist über lange Zeit $r_t = 0$!

$r_t < 0$: negative Verstärkung

Strategie $\pi : \mathcal{S} \rightarrow \mathcal{A}$

Eine Strategie ist optimal, wenn sie langfristig die Belohnung “maximiert”

Abgeschwächte Belohnung

(engl. discounted reward)

Belohnungsfunktion:

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots = \sum_{i=0}^{\infty} \gamma^i r_{t+i}. \quad (10.1)$$

Alternative:

$$V^\pi(s_t) = \lim_{h \rightarrow \infty} \frac{1}{h} \sum_{i=0}^h r_{t+i}. \quad (10.2)$$

Eine Strategie π^* heißt **optimal**, wenn für alle Zustände s

$$V^{\pi^*}(s) \geq V^\pi(s) \quad (10.3)$$

Abkürzung: $V^* = V^{\pi^*}$

Entscheidungsprozesse

Markov-Entscheidungsprozess (engl. Markov decision process, MDP): Belohnung einer Aktion hängt nur von aktuellem Zustand und aktueller Aktion ab

POMDP (engl. partially observable Markov decision process): Zustand des Agenten nicht exakt bekannt.

Uninformierte kombinatorische Suche

Gitter	$\begin{array}{cc} (2) & (2) \end{array}$	$\begin{array}{ccc} (2) & (3) & (2) \\ (3) & (4) & (3) \\ (2) & (3) & (2) \end{array}$	$\begin{array}{cccc} (2) & (3) & (3) & (2) \\ (3) & (4) & (4) & (3) \\ (3) & (4) & (4) & (3) \\ (2) & (3) & (3) & (2) \end{array}$	$\begin{array}{ccccc} (2) & (3) & (3) & (3) & (2) \\ (3) & (4) & (4) & (4) & (3) \\ (3) & (4) & (4) & (4) & (3) \\ (3) & (4) & (4) & (4) & (3) \\ (2) & (3) & (3) & (3) & (2) \end{array}$
Anz. Zust.	4	9	16	25
Strategien	2^4 $= 16$	$2^4 3^4 4$ $= 5184$	$2^4 3^8 4^4$ $\approx 2.7 \cdot 10^7$	$2^4 3^{12} 4^9$ $\approx 2.2 \cdot 10^{12}$

Uninformierte kombinatorische Suche

allgemein:

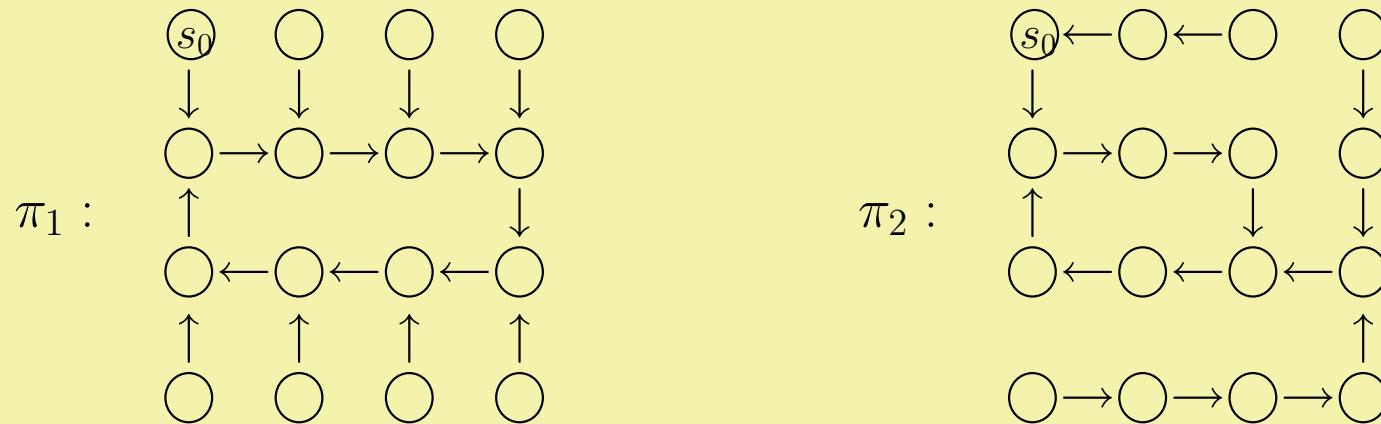
- 4 Eckknoten mit 2 möglichen Aktionen
- $2(n_x - 2) + 2(n_y - 2)$ Randknoten mit 3 Aktionen
- $(n_x - 2)(n_y - 2)$ innere Knoten mit 4 Aktionen

also:

$$2^4 3^{2(n_x-2)+2(n_y-2)} 4^{(n_x-2)(n_y-2)}$$

verschiedene Strategien

Wert von Zuständen



Bewegung nach rechts mit 1 belohnt, nach links mit -1 bestraft

mittlerer Vortrieb für π_1 : $3/8 = 0.375$, mittlerer Vortrieb für π_2 : $2/6 \approx 0.333$

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

γ	0.9	0.8375	0.8
$V^{\pi_1}(s_0)$	2.81	1.38	0.96
$V^{\pi_2}(s_0)$	2.66	1.38	1.00

größeres γ : größerer Zeithorizont für die Bewertung von Strategien!

Wert-Iteration und Dynamische Programmierung

Dynamische Programmierung, Richard Bellman, 1957 Bellman:

Unabhängig vom Startzustand s_t und der ersten Aktion a_t müssen ausgehend von jedem möglichen Nachfolgezustand s_{t+1} alle folgenden Entscheidungen optimal sein.

global optimale Strategie durch lokale Optimierungen

Gesucht ist eine optimale Strategie π^* , welche

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

und

$$V^{\pi^*}(s) \geq V^\pi(s)$$

erfüllt. Es folgt

$$V^*(s_t) = \max_{a_t, a_{t+1}, a_{t+2}, \dots} (r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \dots). \quad (10.4)$$

$r(s_t, a_t)$ hängt nur von s_t und a_t ab, also

$$V^*(s_t) = \max_{a_t} [r(s_t, a_t) + \gamma \max_{a_{t+1}, a_{t+2}, \dots} (r(s_{t+1}, a_{t+1}) + \gamma r(s_{t+2}, a_{t+2}) + \dots)] \quad (10.5)$$

$$= \max_{a_t} [r(s_t, a_t) + \gamma V^*(s_{t+1})]. \quad (10.6)$$

Bellman-Gleichung: (Fixpunktgleichung)

$$V^*(s) = \max_a [r(s, a) + \gamma V^*(\delta(s, a))]. \quad (10.7)$$

also

$$\pi^*(s) = \operatorname{argmax}_a [r(s, a) + \gamma V^*(\delta(s, a))]. \quad (10.8)$$

Iterationsvorschrift: (Fixpunktiteration)

$$\hat{V}(s) = \max_a [r(s, a) + \gamma \hat{V}(\delta(s, a))] \quad (10.9)$$

Initialisierung: $\forall s \quad \hat{V}(s) = 0$

WERT-ITERATION()

For all $s \in \mathcal{S}$

$$\hat{V}(s) = 0$$

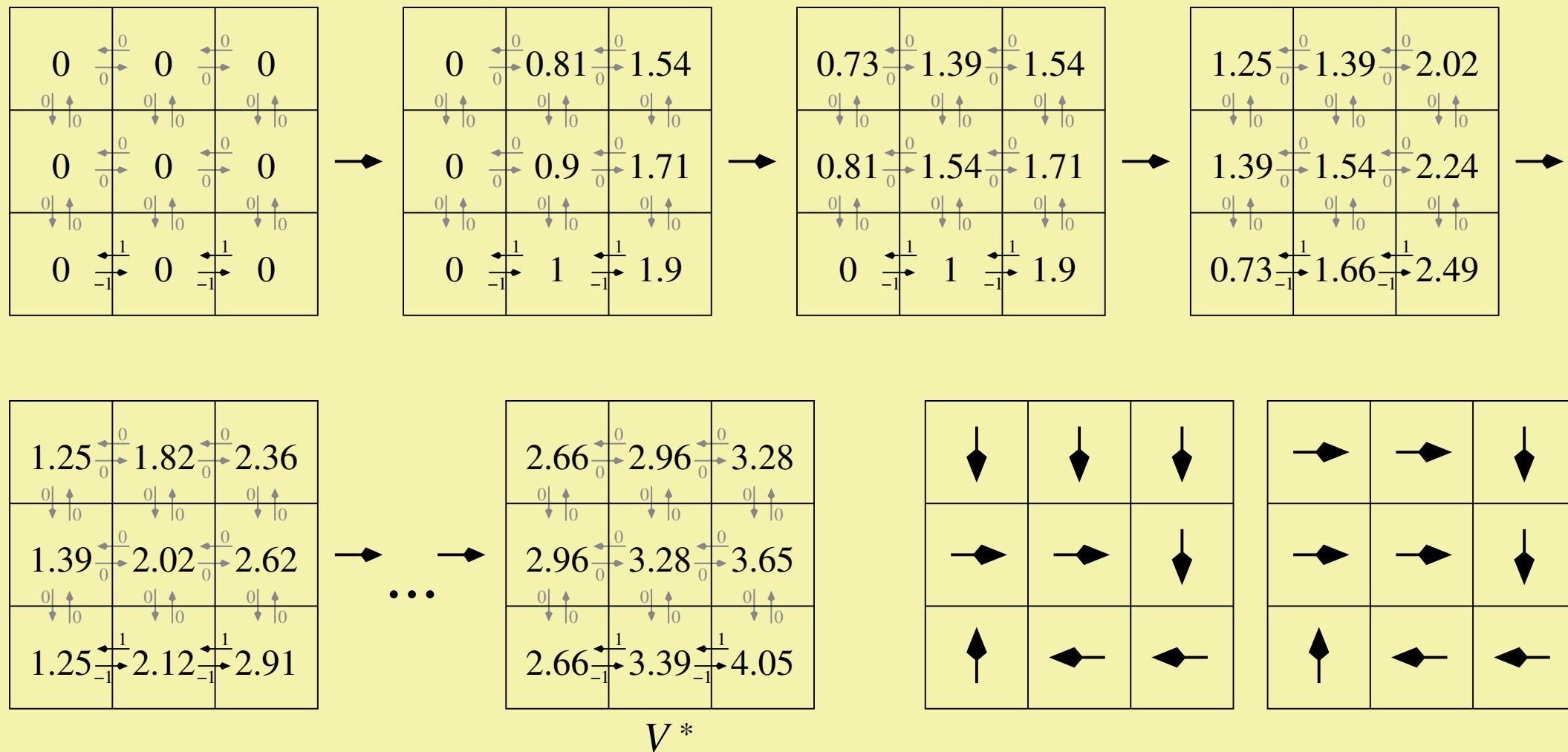
Repeat

For all $s \in \mathcal{S}$

$$\hat{V}(s) = \max_a [r(s, a) + \gamma \hat{V}(\delta(s, a))]$$

Until $\hat{V}(s)$ sich nicht mehr ändert

Satz 10.2 Die Wert-Iteration konvergiert gegen V^* Sutton/Barto.



Wert-Iteration mit $\gamma = 0.9$ und zwei optimale Strategien.

Achtung: es ist falsch, die Aktion zu wählen, welche zum Zustand mit maximalem V^* -Wert führt. Warum?

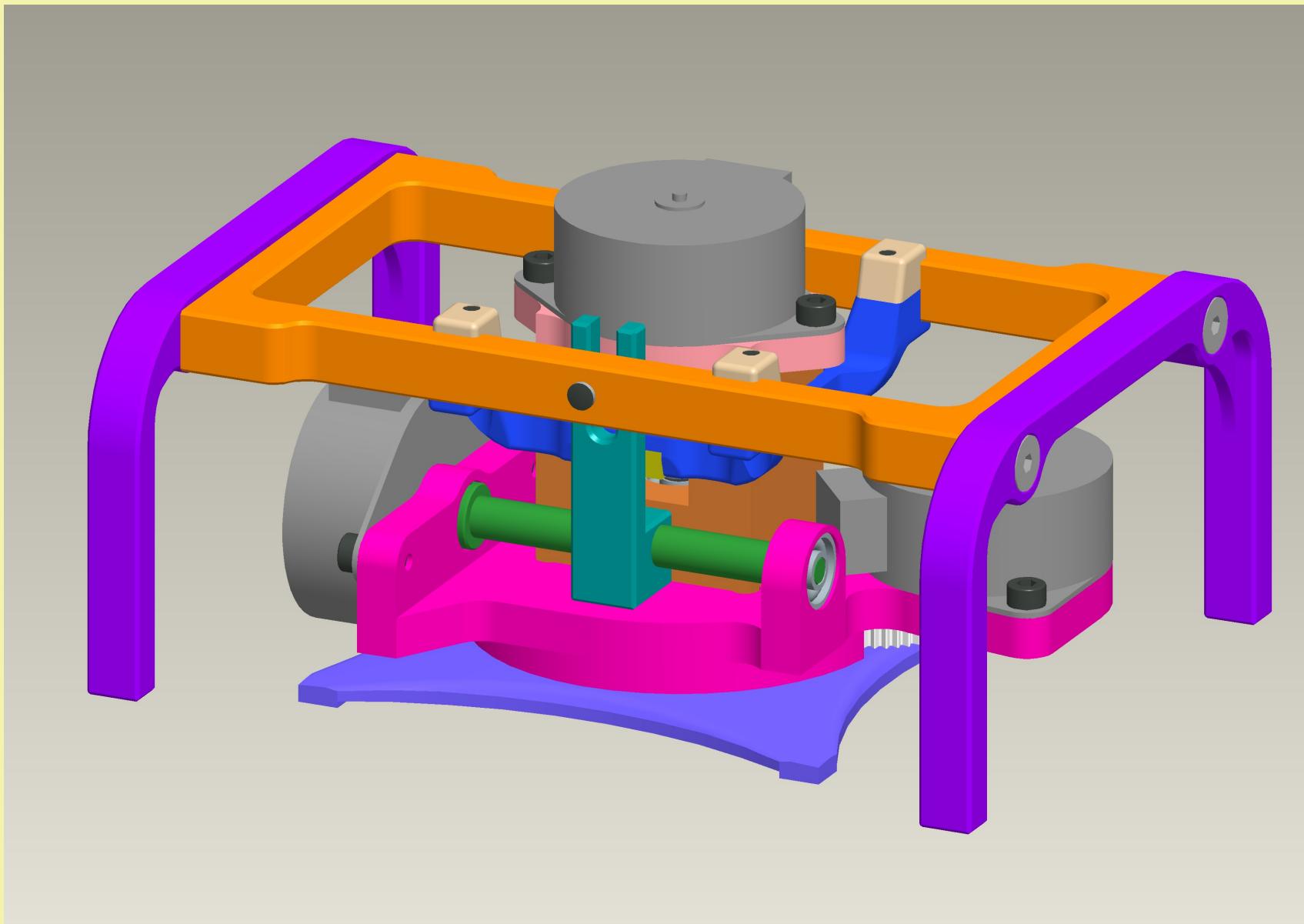
Anwendung auf $s = (2, 3)$ in

2.66	0 0 0 0 0	2.96	0 0 0 0 0	3.28
0 0 0 0 0	↓ ↓ ↓ ↓ ↓	0 0 0 0 0	↑ ↑ ↑ ↑ ↑	0 0 0 0 0
2.96	0 0 0 0 0	3.28	0 0 0 0 0	3.65
0 0 0 0 0	↓ ↓ ↓ ↓ ↓	0 0 0 0 0	↑ ↑ ↑ ↑ ↑	0 0 0 0 0
2.66	-1 1 -1 1 -1	3.39	-1 1 -1 1 -1	4.05

V^*

$$\begin{aligned}
 \pi^\star(2, 3) &= \operatorname{argmax}_{a \in \{\text{links, rechts, oben}\}} [r(s, a) + \gamma V^\star(\delta(s, a))] \\
 &= \operatorname{argmax}_{\{\text{links, rechts, oben}\}} \{1 + 0.9 \cdot 2.66, -1 + 0.9 \cdot 4.05, 0 + 0.9 \cdot 3.28\} \\
 &= \operatorname{argmax}_{\{\text{links, rechts, oben}\}} \{3.39, 2.65, 2.95\} \\
 &= \text{links}
 \end{aligned}$$

Der Laufroboter in Hardware

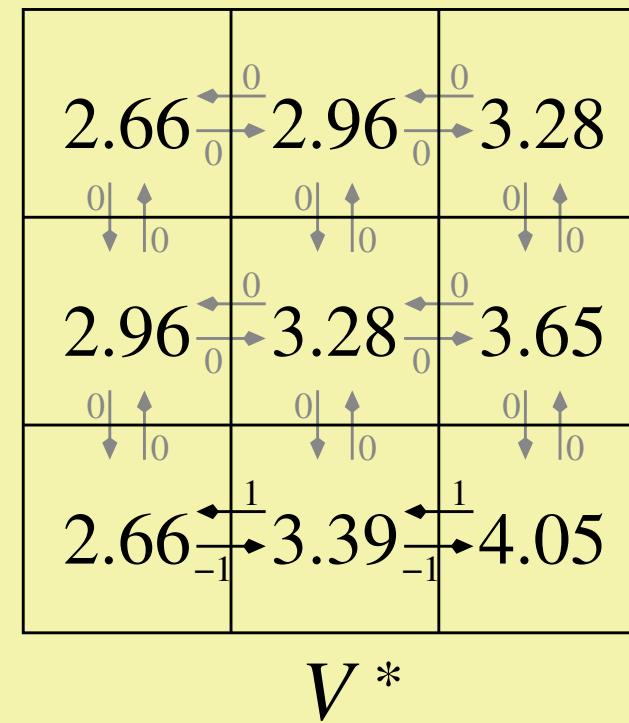


Demo: Laufroboter

Unbekannte Welt

was tun, wenn der Agent kein Modell für seine Aktionen hat?

$$\hat{V}(s) = \max_a [r(s, a) + \gamma \hat{V}(\delta(s, a))]$$



Q-Lernen

Bewertungsfunktion $Q(s_t, a_t)$

$$\pi^*(s) = \operatorname{argmax}_a Q(s, a). \quad (10.10)$$

Abschwächen zukünftiger Belohnungen und maximieren von

$$r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

Bewertung der Aktion a_t im Zustand s_t :

$$Q(s_t, a_t) = \max_{a_{t+1}, a_{t+2}, \dots} (r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \dots). \quad (10.11)$$

Vereinfachung:

$$Q(s_t, a_t) = \max_{a_{t+1}, a_{t+2}, \dots} (r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \dots) \quad (10.12)$$

$$= r(s_t, a_t) + \gamma \max_{a_{t+1}, a_{t+2}, \dots} (r(s_{t+1}, a_{t+1}) + \gamma r(s_{t+2}, a_{t+2}) + \dots) \quad (10.13)$$

$$= r(s_t, a_t) + \gamma \max_{a_{t+1}} (r(s_{t+1}, a_{t+1}) + \gamma \max_{a_{t+2}} (r(s_{t+2}, a_{t+2}) + \dots)) \quad (10.14)$$

$$= r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \quad (10.15)$$

$$= r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(\delta(s_t, a_t), a_{t+1}) \quad (10.16)$$

$$= r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a') \quad (10.17)$$

Fixpunktgleichung wird iterativ gelöst mittels:

$$\hat{Q}(s, a) = r(s, a) + \gamma \max_{a'} \hat{Q}(\delta(s, a), a') \quad (10.18)$$

Q-LERNEN()

For all $s \in \mathcal{S}, a \in \mathcal{A}$

$$\hat{Q}(s, a) = 0 \text{ (oder zufällig)}$$

Repeat

Wähle (z.B. zufällig) einen Zustand s

Repeat

Wähle eine Aktion a und führe sie aus

Erhalte Belohnung r und neuen Zustand s'

$$\hat{Q}(s, a) := r(s, a) + \gamma \max_{a'} \hat{Q}(s', a')$$

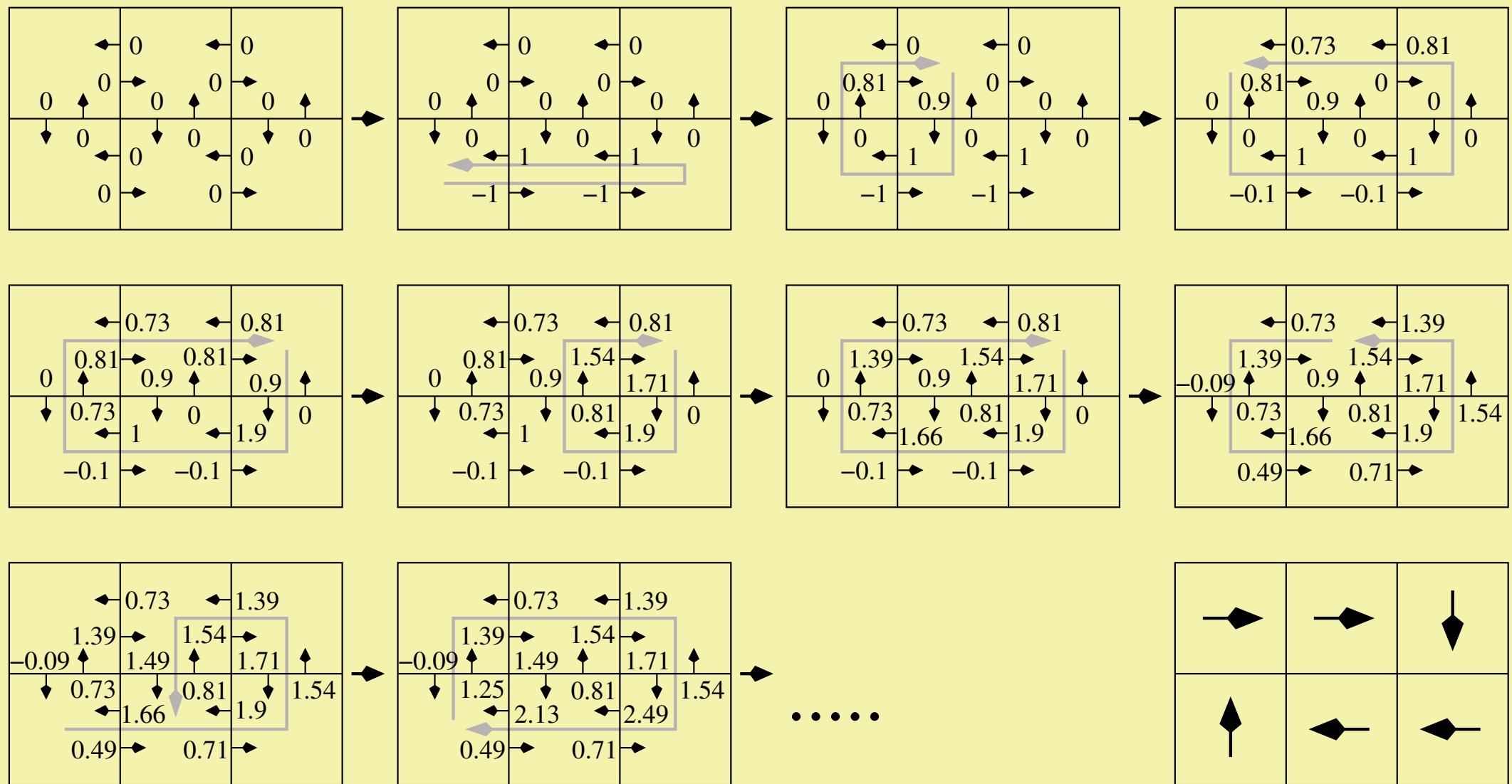
$$s := s'$$

Until s ist ein Endzustand **Oder** Zeitschranke erreicht

Until \hat{Q} konvergiert

Der Algorithmus für das Q-Lernen.

Anwendung des Verfahrens mit $\gamma = 0.9$ und $n_x = 3, n_y = 2$



Satz 10.4 Gegeben sei ein deterministischer MDP mit beschränkten direkten Belohnungen $r(s, a)$. Zum Lernen wird Gleichung 10.18 mit $0 \leq \gamma < 1$ verwendet. Sei $\hat{Q}_n(s, a)$ der Wert für $\hat{Q}(s, a)$ nach n Aktualisierungen. Wird jedes Zustands-Aktions-Paar unendlich oft besucht, so konvergiert $\hat{Q}_n(s, a)$ für alle Werte von s und a gegen $Q(s, a)$ für $n \rightarrow \infty$.

Beweis:

- Jeder Zustands-Aktions-Übergang kommt unendlich oft vor
- Betrachte Intervalle in denen alle Zustands-Aktions-Übergänge mindestens einmal vorkommen.
- Der max. Fehler in der \hat{Q} -Tabelle wird in jedem dieser Intervalle um mindestens den Faktor γ reduziert:

Sei

$$\Delta_n = \max_{s,a} |\hat{Q}_n(s, a) - Q(s, a)|$$

der maximale Fehler in der Tabelle \hat{Q}_n und $s' = \delta(s, a)$.

Für jeden Tabelleneintrag $\hat{Q}_n(s, a)$ gilt:

$$\begin{aligned} |\hat{Q}_{n+1}(s, a) - Q(s, a)| &= |(r + \gamma \max_{a'} \hat{Q}_n(s', a')) - (r + \gamma \max_{a'} Q(s', a'))| \\ &= \gamma |\max_{a'} \hat{Q}_n(s', a') - \max_{a'} Q(s', a')| \\ &\leq \gamma \max_{a'} |\hat{Q}_n(s', a') - Q(s', a')| \\ &\leq \gamma \max_{s'', a'} |\hat{Q}_n(s'', a') - Q(s'', a')| = \gamma \Delta_n. \end{aligned}$$

Die erste Ungleichung gilt, weil für beliebige Funktionen f und g

$$|\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)|$$

und die zweite, weil durch zusätzliches Variieren des Zustandes s'' das resultierende Maximum nicht kleiner werden kann. Es folgt also

$$\Delta_{n+1} \leq \gamma \Delta_n \quad \text{und} \quad \Delta_k \leq \gamma^k \Delta_0 \quad \text{also:} \quad \lim_{n \rightarrow \infty} \Delta_n = 0$$

Bemerkungen

- Q-Lernen konvergiert nach Satz 10.4 **unabhängig** von den während des Lernens gewählten Aktionen.
- Konvergenzgeschwindigkeit ist **abhängig** von den während des Lernens gewählten Aktionen.

Q-Lernen in nichtdeterministischer Umgebung

nichtdeterministischer Agent: Reaktion der Umgebung auf die Aktion a im Zustand s ist nichtdeterministisch.

$$Q(s_t, a_t) = E(r(s, a)) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a'), \quad (10.19)$$

- Konvergenzgarantie für das Q-Lernen geht verloren!
- Grund: bei gleichem Zustand s und gleicher Aktion a völlig verschiedene Reaktion der Umgebung

Neue Lernregel

$$\hat{Q}_n(s, a) = (1 - \alpha_n)\hat{Q}_{n-1}(s, a) + \alpha_n[r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a')] \quad (10.20)$$

mit zeitlich variablem Gewichtungsfaktor

$$\alpha_n = \frac{1}{1 + b_n(s, a)}.$$

- $b_n(s, a)$ gibt an, wie oft bis zur n -ten Iteration im Zustand s die Aktion a schon ausgeführt wurde.
- stabilisierender Term $\hat{Q}_{n-1}(s, a)$.
- Werte $b_n(s, a)$ für alle Zustands-Aktions-Paare müssen gespeichert werden.

TD-Fehler und TD-Lernen

$\alpha_n = \alpha$ (konstant):

$$\begin{aligned}\hat{Q}_n(s, a) &= (1 - \alpha)\hat{Q}_{n-1}(s, a) + \alpha[r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a')] \\ &= \hat{Q}_{n-1}(s, a) + \underbrace{\alpha [r(s, a) + \gamma \max_{a'} \hat{Q}_{n-1}(\delta(s, a), a') - \hat{Q}_{n-1}(s, a)]}_{\text{TD-Fehler}}\end{aligned}$$

- $\alpha = 1$: Q-Lernen
- $\alpha = 0$: Es findet kein Lernen statt
- $0 < \alpha < 1$: ???

Erkunden und Verwerten

Q-LERNEN()

For all $s \in \mathcal{S}, a \in \mathcal{A}$

$$\hat{Q}(s, a) = 0 \text{ (oder zufällig)}$$

Repeat

Wähle (z.B. zufällig) **einen (welchen?) Zustand** s

Repeat

Wähle **eine (welche?) Aktion** a und führe sie aus

Erhalte Belohnung r und neuen Zustand s'

$$\hat{Q}(s, a) := r(s, a) + \gamma \max_{a'} \hat{Q}(s', a')$$

$$s := s'$$

Until s ist ein Endzustand **Oder** Zeitschranke erreicht

Until \hat{Q} konvergiert

Möglichkeiten zur Wahl der nächsten Aktion

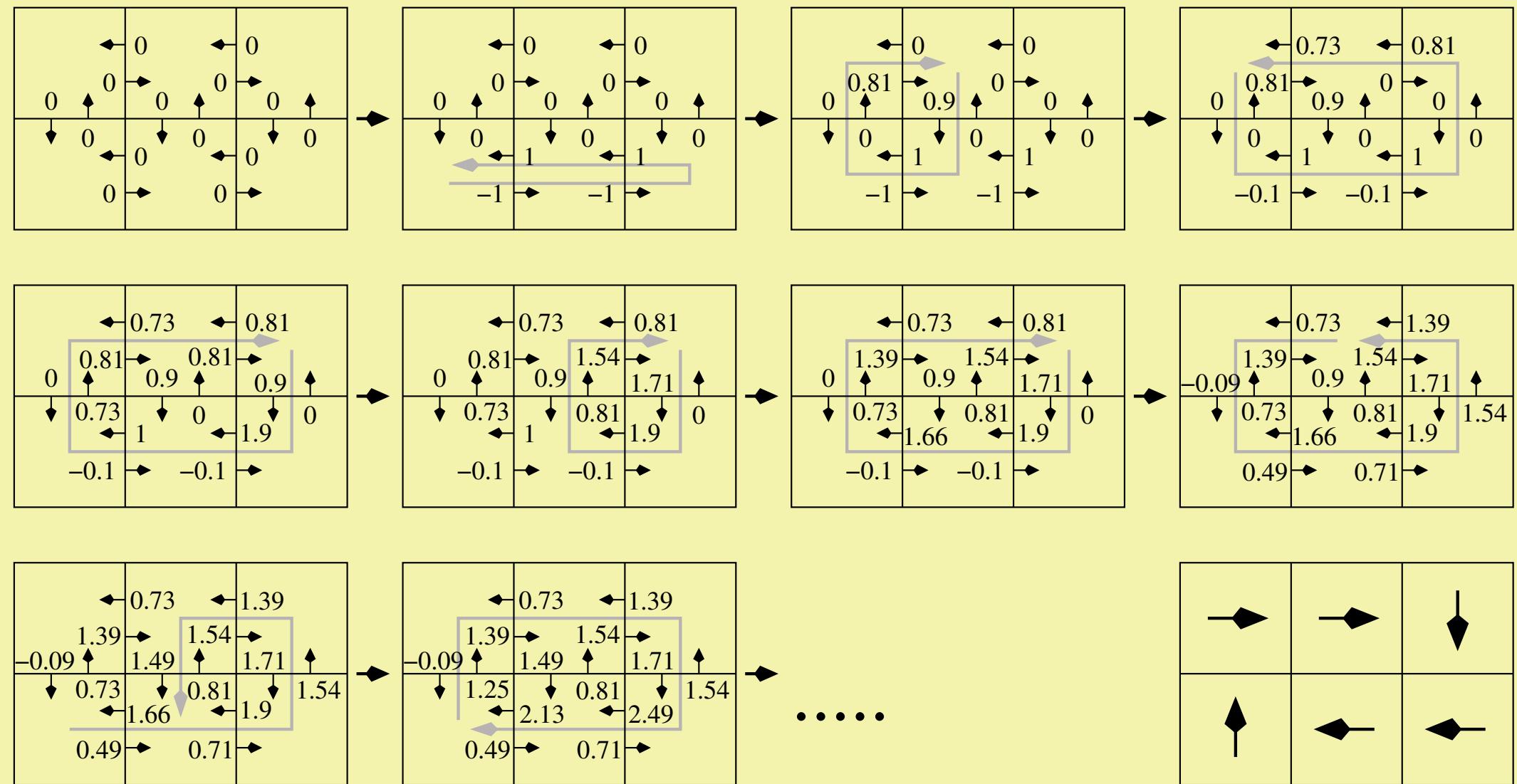
zufällige Wahl:

- führt zu gleichmäßigem **Erkunden (engl. exploration)** aller möglichen Aktionen
- sehr langsame Konvergenz

immer die beste Aktion wählen (höchster \hat{Q} -Wert:)

- optimales **Verwerten (engl. exploitation)** schon gelernten Verhaltens
- relativ schnelle Konvergenz
- nicht optimale Strategien werden gelernt

Wahl des Startzustands



Funktionsapproximation, Generalisierung und Konvergenz

- Stetige Variablen \Rightarrow unendlicher Zustandsraum
- Tabelle mit V - oder Q -Werten kann nicht mehr gespeichert werden

Lösung:

- $Q(s, a)$ -Tabelle wird ersetzt durch ein neuronales Netz mit den
- Input-Variablen s, a und Q -Wert als Ziel-Output.
- Endliche Repräsentation der (unendlichen) Funktion $Q(s, a)$!
- Generalisierung (aus endlich vielen Trainingsbeispielen)

Achtung: Keine Konvergenzgarantie mehr, denn Satz 10.4 gilt nur, wenn jedes Zustands-Aktionspaar unendlich oft besucht wird.

Alternative: beliebiger anderer Funktionsapproximator

POMDP

POMDP (engl. partially observable Markov decision process):

- viele verschiedene Zustände werden als einer erkennt.
- viele Zustände in der realen Welt werden auf eine **Beobachtung** (engl. observation) abgebildet.
- Konvergenzprobleme bei der Wert-Iteration oder beim Q-Lernen
- Lösungsansätze:¹, Observation Based Learning².

¹Sutton, R./Barto, A. Reinforcement Learning. MIT Press, 1998.

²Lauer, M./Riedmiller, M. Generalisation in Reinforcement Learning and the Use of Observation-Based Learning. In Kokai, Gabriella/Zeidler, Jens (Hrsg.) Proceedings of the FGML Workshop 2002. 2002.

Anwendungen: TD-Gammon

- TD-Learning (Temporal Difference Learning) verwendet weiter in der “Zukunft” liegende Zustände
- TD-Gammon: einem Programm zum Spielen von Backgammon
- TD-Learning zusammen mit einem Backpropagation-Netz mit 40 bis 80 verdeckten Neuronen
- Einzige direkte Belohnung: Ergebnis am Ende eines Spiels.
- TD-Gammon wurde trainiert in 1.5 Millionen Spielen gegen sich selbst.
- Es besiegte Weltklassespieler.

Weitere Anwendungen

- RoboCup: mit Lernen durch Verstärkung wird heute das Verhalten der Roboter gelernt, z.B. Dribbeln³.
- Inverses Pendel
- Steuerung eines Quadrocopter

Probleme in der Robotik:

- Extreme Rechenzeiten bei hochdimensionalen Problemen (viele Variablen/Aktionen)
- Feedback der Umwelt bei realen Robotern ist sehr langsam.
- Bessere, schnellere Lernalgorithmen werden benötigt.

³Stone, P./Sutton, R.S./Kuhlmann, G. Reinforcement Learning for RoboCup-Soccer Keepaway. *Adaptive Behavior*, 2005; The RoboCup Soccer Simulator. <http://sserver.sourceforge.net>.

Landen von Flugzeugen



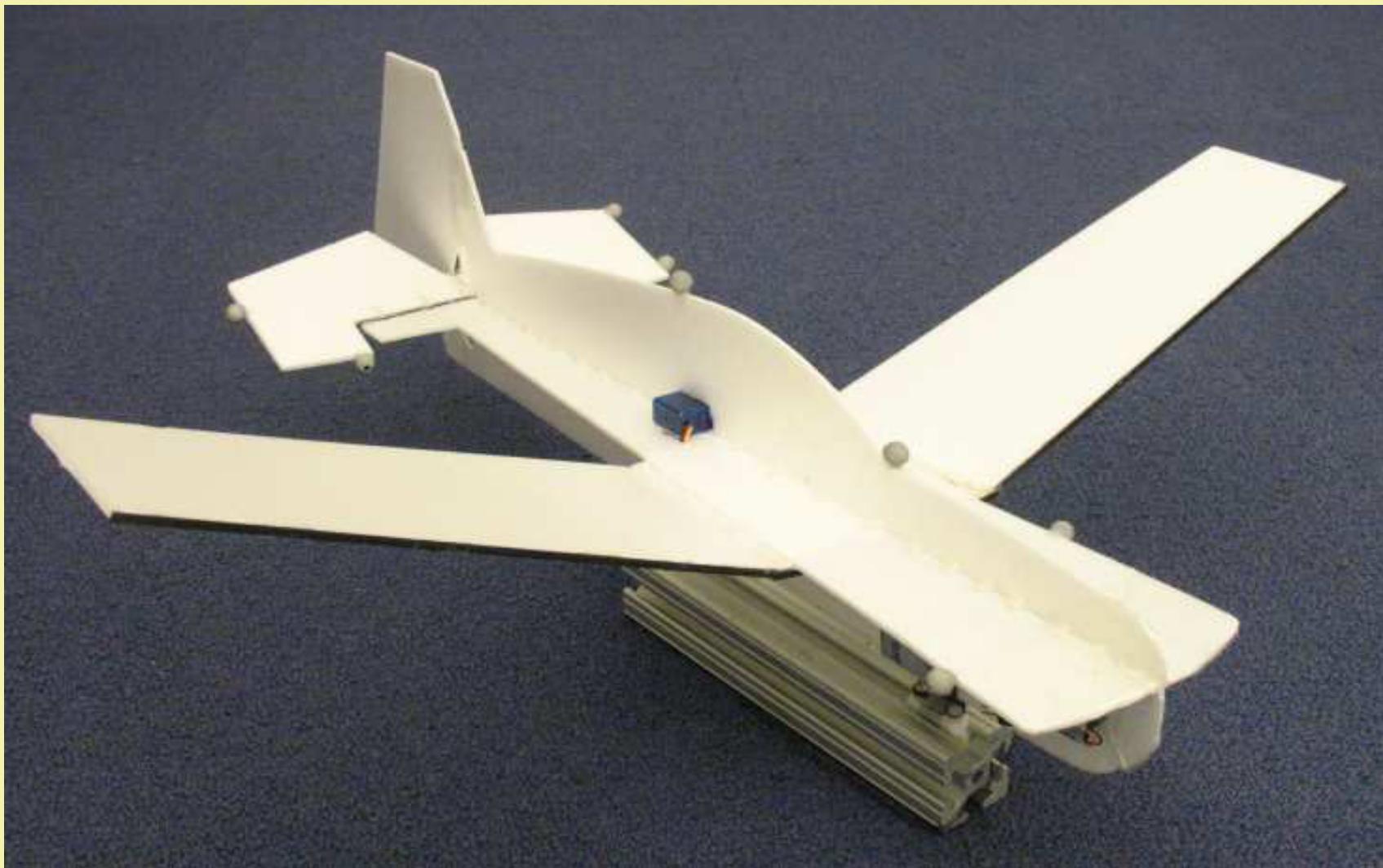
[Russ Tedrake, IROS 08]

Birds don't solve Navier-Stokes!



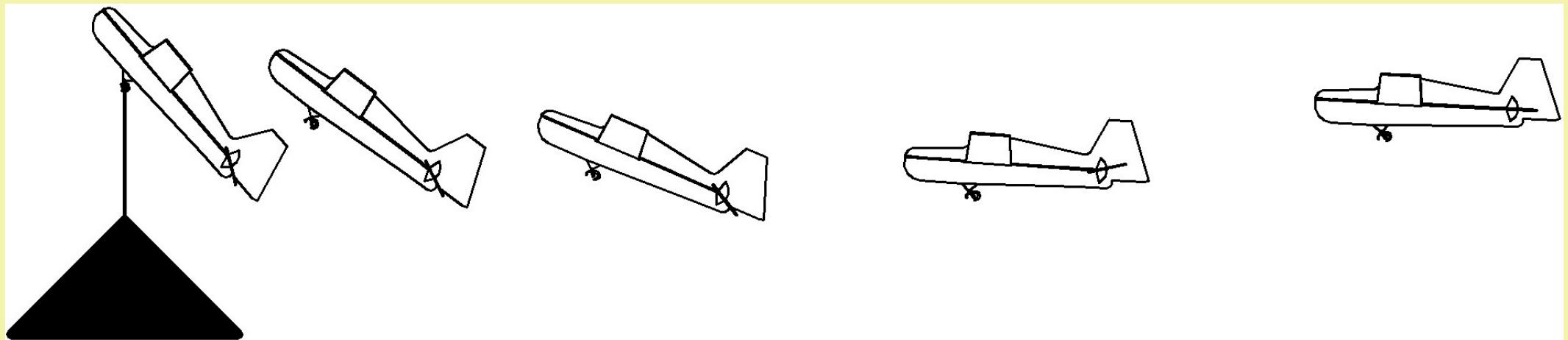
[Russ Tedrake, IROS 08]

Birds don't solve Navier-Stokes!



[Russ Tedrake, IROS 08]

Birds don't solve Navier-Stokes!



[Russ Tedrake, IROS 08]

Fluch der Dimensionen

curse of dimensionality

- Problem: hochdimensionale Zustands- und Aktionsräume

Lösungsmöglichkeiten:

- Lernen in der Natur auf vielen Abstraktionsebenen
- Informatik: jede gelernte Fähigkeit wird in ein Modul gekapselt
- Aktionsraum wird stark verkleinert
- Zustände werden abstrahiert
- hierarchisches Lernen [Barto/Mahadevan](#)
- verteiltes Lernen (Tausendfüßler ein Gehirn pro Bein)

Fluch der Dimensionen, weitere Ideen

- Menschl. Gehirn ist bei Geburt keine Tabula Rasa
- Gute initiale Strategie für Roboter?
 - 1. Klassische Programmierung.
 - 2. Lernen durch Verstärkung
 - 3. Trainer gibt zusätzliches Feedback
- oder:
 - 1. Lernen durch Demonstration (lernen mit Lehrer)
 - 2. Lernen durch Verstärkung Billard et al.
 - 3. Trainer gibt zusätzliches Feedback

Aktuelle Forschung

- Fitted Value Iteration
- Verbindung von Lernen durch Verstärkung mit Imitation Learning
- Policy Gradienten Methoden
- Actor Critic Methoden
- Natural Gradient Methoden

Fitted Value Iteration

```
1: Randomly sample  $m$  states from the MDP
2:  $\Psi \leftarrow 0$ 
3:  $n \leftarrow$  the number of available actions in  $A$ 
4: repeat
5:   for  $i = 1 \rightarrow m$  do
6:     for  $j = 1 \rightarrow n$  do
7:        $q(a) \leftarrow R(s^{(i)}) + \gamma V(s^{(j)})$ 
8:     end for
9:      $y^{(i)} \leftarrow \max q(a)$ 
10:   end for
11:    $\Psi \leftarrow \arg \min_{\Psi} \sum_{i=1}^m (y^{(i)} - \Psi^T \Phi(s^{(i)}))^2$ 
12: until  $\Psi$  Converges
```

- reinforcement learning algorithms:
 - value iteration
 - $Q(\lambda)$, SARSA(λ)
 - TD(λ)
 - tabular and function approximation versions
 - actor critic
 - tile coding
 - locally weighted regression
- Example Environments:
 - mountain car
 - gridworld (with editor), windy gridworld
 - dicegame
 - n armed bandit
 - pole swing up

Literatur

- Erste Einführung: [Mitchell, T.](#) Machine Learning. McGraw Hill, 1997
- Standardwerk: [Sutton, R./Barto, A.](#) Reinforcement Learning. MIT Press, 1998
- Übersicht: [Kaelbling, L.P./Littman, M.L./Moore, A.P.](#) Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, 4 1996