Logikprogrammierung, Stetigkeit, künstliche neuronale Netzwerke

Pascal Hitzler

BIGS/Oberseminar Mengenlehre, Bonn, Februar 2002

Slide 1

Motivation

- *Biologische* neuronale Netzwerke können (leicht) logische Schlussfolgerungen ziehen.
- Warum ist das mit *künstlichen* so schwierig?

Institut für künstliche Intelligenz, Fakultät für Informatik,
Technische Universität Dresden
phitzler@inf.tu-dresden.de http://www.wv.inf.tu-dresden.de/~pascal/

Slide 2

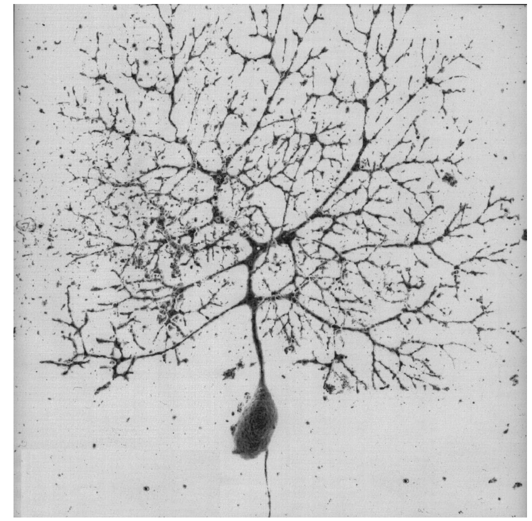
Inhalt

- Logik und künstliche neuronale Netzwerke.
- Stetigkeit semantischer Operatoren in der Logikprogrammierung.
- Logikprogrammierung, Stetigkeit, künstliche neuronale Netzwerke.

Dank für Diskussionen an:
Howard A. **Blair** (Syracuse), Steffen **Hölldobler**, Hans-Peter **Störr** (Dresden).
Neue Resultate in Zusammenarbeit mit Anthony Karel **Seda** (Cork).

Slide 3

Biologische neuronale Netze



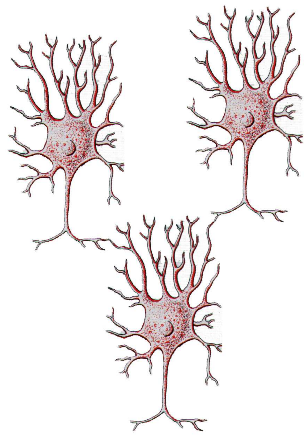
Neuron, mit Dendriten, Soma und Axon.

(Purkinjezelle im Cerebellum)

Bild: Spektrum der Wissenschaft 10, Oktober 2001

Biologische neuronale Netze

Slide 4



Potenziale werden von den Dendriten zum Soma propagiert.

Wenn das akkumulierte Potenzial einen Schwellenwert überschreitet, feuert das Neuron.

Das resultierende Potenzial wird entlang des Axons zu weiteren Neuronen propagiert.

Bilder: Birbaumer & Schmidt, Biologische Psychologie, ^{2}1991

Künstliche neuronale Netze

Slide 5

(Endliche) Menge von *Einheiten* (Knoten, Neuronen) mit *Verbindungen*.
▶ Graph

- Potenziale sind reelle Zahlen.
- Erregungsleitung benötigt keine Zeit.
- Potenziale akkumulieren als gewichtete Summen.
(Gewichte stehen für Synapsenaktivität und können „gelernt“ werden.)
- Einheiten werden in diskreten Zeitschritten aktiv.
- Schwellenfunktion ist dieselbe im ganzen Netzwerk.

Es existieren viele konkurrierende Architekturen.

Künstliche neuronale Netze

Slide 6

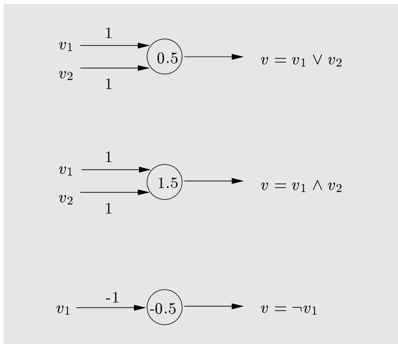
Insbesondere:

- Jede Einheit berechnet eine *simple* Eingabe-Ausgabefunktion.
- Die Einheiten sind *blind* bezüglich dessen, woher ihre Eingabe kommt und wohin ihre Ausgabe geht.

Information (Wissen) wird durch die
(*gewichteten*) *Verbindungen*
im Netzwerk repräsentiert!

▶ *Konnektionistische Systeme.*

McCulloch-Pitts Netzwerke



Slide 7

McCulloch & Pitts 1943

Neuronen mit binärer Schwellenfunktion für $\vee$, $\wedge$, $\neg$.

Updates werden für alle Einheiten synchron berechnet.

McCulloch-Pitts Netzwerke sind genau die endlichen Automaten.

Bild: Hölldobler, Vorlesungsskript *Introduction to Computational Logic*, 2001

McCulloch-Pitts Netzwerke: Weiterentwicklungen

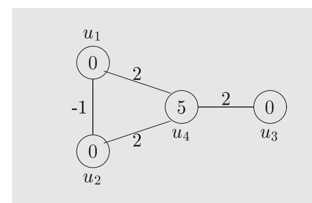
Slide 8

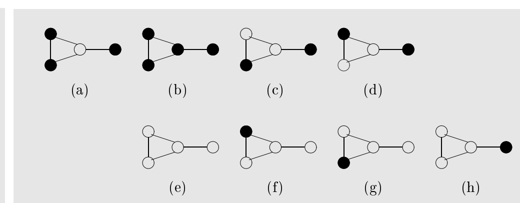
Hölldober & Kalinke 1994: Repräsentation aussagenlogischer Programme durch dreischichtige vorwärtsgerichtete Netze.

d'Avila Garcez, Broda, Gabbay, Zaverucha 1999/2001:
Erweiterung auf sigmoidale (differenzierbare) Schwellenfunktionen.
Lernen durch Rückpropagierung möglich (eine Art Gradientenabstieg).

Idee:
- ▶ Wissensrepräsentation durch Netwerk
  - ▶ Lernen durch Rückpropagierung
    - ▶ Extraktion des erlernten Wissens

Symmetrische Netze und Aussagenlogik





Slide 9

Pinkas 199x: Hopfield-Netzwerke mit symmetrischen Verbindungen.

Update asynchron durch zufällige Auswahl je einer Einheit.

Es ex. ein Zusammenhang zwischen stabilen Zuständen im Netzwerk und Modellen aussagenlogischer Formeln (via Energieminimierung).

▶ Nichtmonotone Aussagenlogik.
▶ Stochastische Netzwerke.

Bilder: Hölldobler, Vorlesungsskript *Introduction to Computational Logic*, 2001

Jenseits der Aussagenlogik

Slide 10

Variablenbindungen?

Termrepräsentationen?

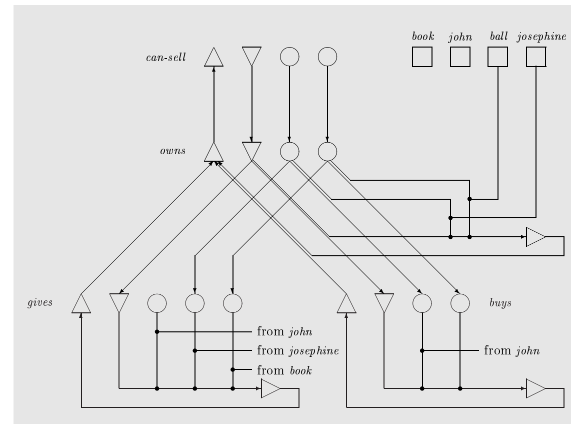
Unendliche Grundinstanzierungen?

SHRUTI

Slide 11

Variablenbindung durch zeitliche Synchronisierung.

gives(X,Y,Z)→ owns(Y,Z)	gives(john,josephine,book)
buys(X,Y)→ owns(X,Y)	(∃X) buys(john,X)
owns(X,Y)→ can-sell(X,Y)	owns(josephine,ball)



Reflexive Reasoning mit diesem System möglich.

Bild: Hölldobler, Vorlesungsskript *Introduction to Computational Logic*, 2001

Slide 12

SHRUTI

Regeln: $\forall(p_1(\dots) \wedge \dots \wedge p_n(\dots) \rightarrow (\exists Y_1, \dots, Y_k) p(\dots))$

Fakten und Anfragen: $\exists(q(\dots))$

Einige Einschränkungen:

- Keine Funktionssymbole außer Konstanten.
- In Fakten oder Anfragen kommt jede Variable höchstens einmal vor.
- Jede Variable, die in der Bedingung einer Regel mehr als einmal vorkommt, muss auch in deren Konsequenz auftauchen. Außerdem muss sie gebunden sein, wenn die Konsequenz mit einer Anfrage unifiziert wird.
- Die Anzahl der Anwendungen derselben Regel bei einer Ableitung ist global beschränkt.

Slide 13

Hebbsches Lernen

Ashish Darbari, 2000 (Diplomarbeit, Dresden)

Kombiniert Variablenbindung durch zeitliche Synchronisierung mit Hebbschem Lernen.

Slide 14

Logikprogramme

Ein (*normales logisches*) *Programm* P ist eine endliche Menge von *Klauseln*

$$\forall(A \leftarrow L_1, \dots, L_n),$$

über einer Sprache erster Stufe, abgekürzt als

$$A \leftarrow L_1, \dots, L_n,$$

wobei A ein Atom und die L_i Literale sind. (A *Kopf*, $L_1, \dots, L_n$ *Rumpf*.)

P heißt *definit*, wenn P keine Negation enthält.

B_P: Menge aller Grundinstanzen von Atomen
(hier: nur Herbrand).
$I_P = 2^{B_P}$: Menge aller Interpretationen (vollständiger Verband bzgl. $\subseteq$).
ground(P): Menge aller Grundinstanzen von Klauseln in P.

Slide 15

Fixpunktsemantik

P Programm. Definiere $T_P : I_P \rightarrow I_P$ durch:

$T_P(I)$ ist Menge aller $A \in B_P$ für welche eine Klausel $A \leftarrow L_1, \dots, L_n$ in ground(P) existiert mit $I \models L_1 \wedge \dots \wedge L_n$.

Eigenschaften des *Konsequenzoperators* T_P:

- Modelle von P sind die Prä-Fixpunkte von T_P.
(Fixpunkte von T_P heißen *unterstützte Modelle*.)
- T_P ist *monoton* und *Scott-stetig* wenn P definit ist.
- T_P ist i.A. nicht monoton wenn P nicht definit ist.

Scott-Stetigkeit

P definites Programm. Dann:
- $T_P(I) \subseteq T_P(J)$ für alle $I \subseteq J$ (Monotonie).
- $\operatorname{lub} T_P(I_n) = T_P(\operatorname{lub} I_n)$ für alle gerichteten (I_n).

D.h. T_P *Scott-stetig.* Subbasis der Scott-Topologie auf I_P:

$\{\mathcal{G}(A) : A \in B_P\}$ mit $\mathcal{G}(A) = \{I \in I_P : I \models A\}$.

Slide 16

Satz

> X vollständiger Verband (Scott-Ershov Domain).
> $f : X \to X$ Scott-stetig.
> Dann hat f einen kleinsten Fixpunkt.

Der kleinste Fixpunkt von T_P (Herbrand-Fall) gibt denotationelle Semantik von P.

Negation: Atomtopologie

P nicht definit, dann T_P i.A. nicht monoton (also nicht Scott-stetig).

Fixpunktsatz nicht anwendbar.

Slide 17

$\mathcal{G} = \{\mathcal{G}(A) : A \in B_P\} \cup \{\mathcal{G}(\neg A) : A \in B_P\}$ mit $\mathcal{G}(L) = \{I \in I_P : I \models L\}$
Subbasis der *Atomtopologie* Q auf I_P.

B_P abzählbar dann (I_P, Q) homöomorph zur *Cantor-Menge* $\mathcal{C}$ auf $\mathbb{R}$.

(Batarekh & Subrahmanian 1989, Seda 1995)

Ergebnisse

$\lim T_P^n(I)$ ist Modell von P, falls existent.
(Hitzler & Seda 1997)

Slide 18

$\lim T_P^n(I)$ ist unterstütztes Modell von P, falls existent und T_P stetig ist.

Stetigkeitscharakterisierung
(Seda 1995)

Weiterentwicklungen

Anwendungen (von Verallgemeinerungen) des Banachschen Fixpunktsatzes.

(Hitzler & Seda, Theoretical Computer Science, to appear)
(Prieß-Crampe & Ribenboim 2000)

Slide 19

Fixpunktsemantik nichtmonotoner Logik.

(Hitzler & Seda 1999)

Operatoren in mehrwertiger Logik.

(s. auch Clifford & Seda 2000 (Cork), Heinze 2002 (Bonn))

Mehrwertige Logik

Slide 20

Wahrheitswerte $\mathcal{T} = \{t_1, \dots, t_n\}$.

Interpretationen sind Funktionen $I : B_P \to \mathcal{T}$.

$I_{P,n} = I_P$ Menge aller Interpretationen.

$\mathcal{B}_A$ Menge aller Atome in Klauseln von ground(P) mit Kopf $A \in B_P$.

$T : I_P \to I_P$ *Konsequenzoperator* für P, wenn für jedes $I \in I_P$ und jedes $A \leftarrow \texttt{body}$ in P gilt: $T(I)(A) \leftarrow I(\texttt{body})$ ist wahr nach Wahrheitstafel.

T ist *lokal*, wenn $T(I)(A) = T(K)(A)$ für alle $A \in B_P$ und alle $I, K \in I_P$, die auf $\mathcal{B}_A$ übereinstimmen.

T_P ist ein lokaler Konsequenzoperator.

Andere Beispiele: Operatoren nach Fitting (1985,199x) in drei- bzw. vierwertiger Logik.

Verallgemeinerte Atomtopologie $\mathcal{Q}$

Slide 21

$\mathcal{Q}$ ist die Produkttopologie auf $I_P = \mathcal{T}^{B_P}$, wobei $\mathcal{T} = \{t_1, \dots, t_n\}$ die diskrete Topologie trägt.

$\mathcal{Q}$ ist eine total unzusammenhängende, kompakte und in sich dichte Hausdorfftopologie mit abzählbarer Basis.

$\mathcal{Q}$ ist metrisierbar und homöomorph zur Cantortopologie auf dem Einheitsintervall der reellen Zahlen (B_P ist abzählbar).

Stetigkeit in $\mathcal{Q}$

Slide 22

Konsequenzoperator T auf I_P ist *lokalendlich*, falls für alle $A \in B_P$ und all $I \in I_P$ eine endliche Menge $S \subseteq \mathcal{B}_A$ existiert mit $T(J)(A) = T(I)(A)$ für alle $J \in I_P$, die mit I auf S übereinstimmen.

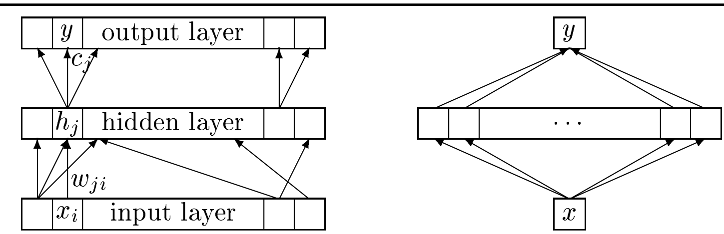
Satz

> Ein lokaler Konsequenzoperator ist genau dann lokalendlich, wenn er stetig in $\mathcal{Q}$ ist.

Hinreichende Bedingungen:
- P enthält keine lokalen Variablen.
- Es gibt eine injektive *Stufenfunktion* $l : B_P \to \mathbb{N}$ mit der Eigenschaft, dass für jedes $A \in B_P$ ein $n_A \in \mathbb{N}$ existiert mit $l(B) < n_A$ für alle $B \in \mathcal{B}_A$. (Mitteilung von H.A. Blair.)

3-schichtige vorwärtsgerichtete Netze (3svN)

Slide 23



x_i Eingabewerte; y Ausgabewert; w_{ji}, c Gewichte der Verbindungen

Eingabe-Ausgabefunktion:

$$y = f(x_1, \dots, x_r) = \sum_j c_j \phi \left(\sum_i w_{ji} x_i - \theta_j \right)$$

θ_j Schwellenwerte

σ Schwellenwertfunktion (sigmoidal oder Gaußsche Glockenkurve).

Slide 24

Idee
(Hölldobler, Kalinke, Störr 1994/1999)

Repräsentation eines Logikprogramms P durch seinen Konsequenzoperator T_P.

Liefert:
Repräsentation eines Programms über einem unendlichen Grundbereich als Grundinstanz eines Logikprogramms in der Prädikatenlogik,
d.h. einer Formel in Klauselform.

Slide 25

Hölldober, Kalinke & Störr 1999

Suche: geeignete Einbettung von I_P in $\mathbb{R}$.

Hölldobler, Kaline & Störr 1999: Spezialfall für azyklische Programme mit injektiver Stufenfunktion.

Via: Metrik aus Fitting (1994) und Darstellung reeller Zahlen im 4er-System.

messy

Außerdem:
Approximation statt Repräsentation.
Reine Existenzaussage (das Netzwerk wird nicht konstruiert).

Slide 26

Stetigkeit

Theorem (Funahashi 1989, vereinfachte Version):

ϕ sigmoidal
$K \subseteq \mathbb{R}$ kompakt,
$f : K \to \mathbb{R}$ stetig,
$\varepsilon > 0$.

Dann gibt es ein 3svN mit Schwellenfunktion ϕ und Eingabe-Ausgabefunktion $\bar{f} : K \to \mathbb{R}$ mit

$$\max_{x \in K} \left\{ d\left(f(x), \bar{f}(x)\right) \right\} < \varepsilon;$$

d Metrik, die die natürliche Topologie auf $\mathbb{R}$ induziert.

D.h. jede stetige Funktion $f : K \to \mathbb{R}$ kann gleichmäßig durch Eingabe-Ausgabefunktionen von 3svNs approximiert werden.

Slide 27

Approximation stetiger Konsequenzoperatoren

Theorem (Hitzler & Seda 2001)

Sei P ein Logikprogramm, T ein lokalendlicher Konsequenzoperator und ι ein Homöomorphismus von $(I_{P,n}, \mathcal{Q})$ nach $\mathcal{C}$.
Dann kann $\iota(T)$ gleichmäßig durch Eingabe-Ausgabefunktionen von 3-schichtigen vorwärtsgerichteten Netzen approximiert werden.

Gleiches gilt z.B. für die Gaußsche Glockenfunktion als Schwellenfunktion.

Messbarkeit

Satz (Hornik, Stinchcombe, White 1989, vereinfachte Version)

$\phi : \mathbb{R} \to (0,1)$ monoton steigend, surjektiv.
$f : \mathbb{R} \to \mathbb{R}$ Borel-messbar,
μ Borel-Wahrscheinlichkeitsmaß auf $\mathbb{R}$,
$\varepsilon > 0$.

Slide 28 Dann gibt es ein 3svN mit Schwellenfunktion ϕ und Eingabe-Ausgabefunktion $\bar{f} : \mathbb{R} \to \mathbb{R}$ mit

$$\varrho_\mu(f, \bar{f}) = \inf \left\{ \delta > 0 : \mu \left\{ x : \left| f(x) - \bar{f}(x) \right| > \delta \right\} < \delta \right\} < \varepsilon.$$

D.h. die Menge der Eingabe-Ausgabefunktionen, die durch 3svNs berechnet werden kann, ist dicht bezüglich ϱ_μ in der Menge aller Borel-messbaren Funktionen $f : \mathbb{R} \to \mathbb{R}$.

Messbare Konsequenzoperatoren

Satz (Hitzler & Seda 2000)

Lokale Konsequenzoperatoren sind immer messbar bezüglich $\sigma(\mathcal{Q})$.

Slide 29

Die Approximation durch 3svNs ist nur *fast überall*.

Die Cantormenge hat Maß 0.

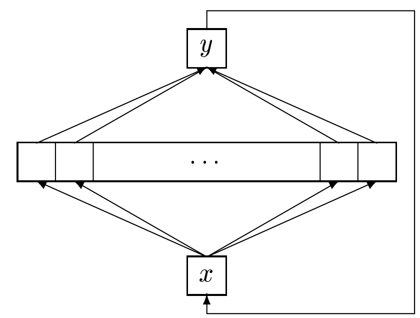
Rekursive Architektur

Die Resultate liefern keinen Hinweis, wie ein approximierendes Netz konstruiert werden könnte.

Dennoch können wir weitere Resultate über das Verhalten solcher Netze ableiten.

Hölldobler, Störr und Kalinke 1999 verwenden folgende Architektur.

Slide 30



Das Netzwerk iteriert den Konsequenzoperator.

Rekursive Architektur

Slide 31

T lokalendlicher Konsequenzoperator.
f Eingabe-Ausgabefunktion des approximierenden Netzes.

Für jedes $I \in I_P$ und jedes $n \in \mathbb{N}$ ist

$$|f^n(\iota(I)) - \iota(T^n(I))| \leq \varepsilon \frac{1-\lambda^n}{1-\lambda}.$$

λ Lipschitz-Konstante von F, der Erweiterung von $\iota(T)$ auf $[0,1]$.
ε Approximationsfehler.

Rekursive Architektur

Slide 32

Ist F eine Kontraktion auf $[0,1]$, dann konvergiert $\left(F^k(\iota(I))\right)$ für jedes I gegen den eindeutigen Fixpunkt x von F und es gibt ein $m \in \mathbb{N}$, so dass für alle $n \geq m$ gilt:

$$|f^n(\iota(I)) - x| \leq \varepsilon \frac{1}{1-\lambda}.$$

Außerdem ist T dann eine Kontraktion auf dem vollständigen Raum I_P (mit geeigneter Metrik), und für den eindeutigen Fixpunkt M von T gilt $\iota(M) = x$.

Rekursive Architektur

Slide 33

Angenommen, es existiert ein $I \in I_P$, so dass $T^n(I)$ in $\mathcal{Q}$ gegen einen Fixpunkt M von T konvergiert.

Dann existiert zu jedem $\delta > 0$ ein $n \in \mathbb{N}$ und ein Netwerk mit $|f^n(\iota(I)) - \iota(M)| < \delta$.

Azyklische Programme

Slide 34

Ein Logikprogramm P ist *azyklisch*, wenn es eine Stufenfunktion $l : B_P \to \mathbb{N}$ gibt, so dass für jedes $A \in B_P$ und jedes $B \in \mathcal{B}_A$ gilt: $l(A) > l(B)$.

Sei $d : I_P \times I_P \to \mathbb{R}$ definiert durch $d(I,J) = 2^n$, wobei n die kleinste Zahl ist, für die I und J sich für ein Atom der Stufe n unterscheiden.

d ist eine vollständige Metrik auf I_P.

Sei P ein azyklisches Programm und T ein lokaler Konsequenzoperator für P. Dann ist T eine Kontraktion bezüglich d und $T^n(I)$ konvergiert in $\mathcal{Q}$ für alle $I \in I_P$ gegen den eindeutigen Fixpunkt von T.

Allgemeinere Programme

Slide 35 Für sehr viel allgemeinere Programme sind Metriken bekannt bezüglich derer T_P eine Kontraktion ist. Diese Metriken sind i.A. aber nicht in die reellen Zahlen einbettbar.

Quo Vadis?

Slide 36 Für azyklische Programme mit injektiver Stufenfunktion sollte es möglich sein, ein approximierendes Netzwerk aus dem Programm zu konstruieren.

Quo Vadis?

Slide 37

Repräsentation von Logik in neuronalen Netzen.

Lernen mit neuronalen Netzen.

Wissensextraktion aus neuronalen Netzen.

Induktive Logikprogrammierung.

Das Diagramm wird sich wohl nicht schließen, aber wir verstehen die Zusammenhänge noch nicht.

Quo Vadis?

Slide 38

Die Repräsentation (Approximation) von Logikprogrammen durch neuronale Netzwerke ist eine Art „Fuzzyfizierung“ (Blair: *continualization*).

Eine andere Art der „Fuzzyfizierung“ ist die quantitative Logikprogrammierung (z.B. van Emden 1986, Mateis 1999).

Gibt es eine „Metatheorie“?