# Logic and Neural Networks

**Pascal Hitzler**

**Indonesia, December 2001**

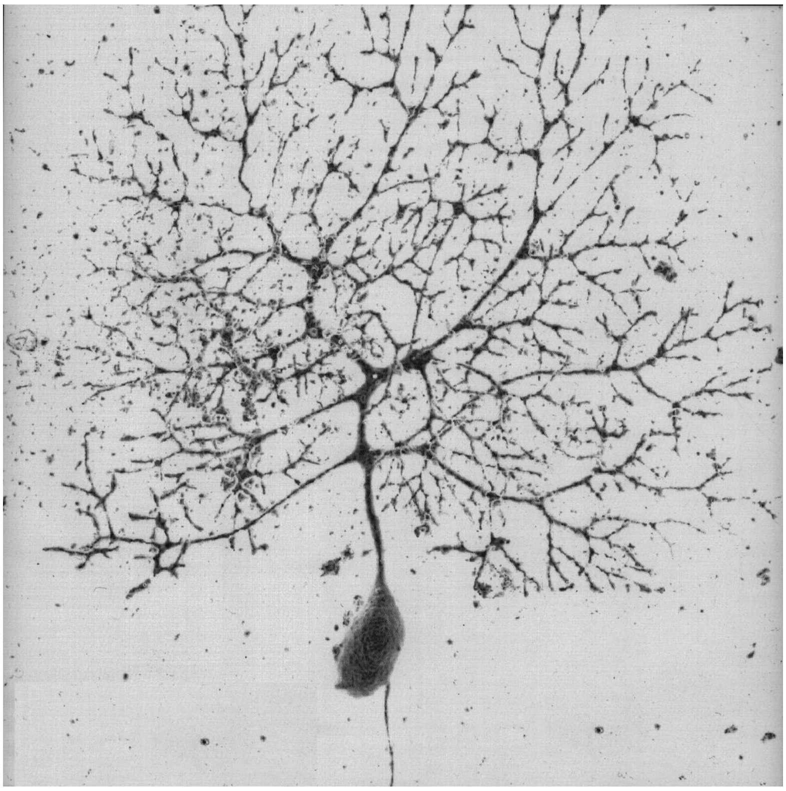
We study mathematical models
for neural networks

and how they relate to logic.

In particular:
How to represent logic by neural networks.

**Dr. Pascal Hitzler**
Knowledge Representation and Reasoning Group
Artificial Intelligence Institute
Department of Computer Science
**Dresden** University of Technology, **Germany**
phitzler@inf.tu-dresden.de
http://www.wv.inf.tu-dresden.de/∼pascal/

# Biological Neural Networks



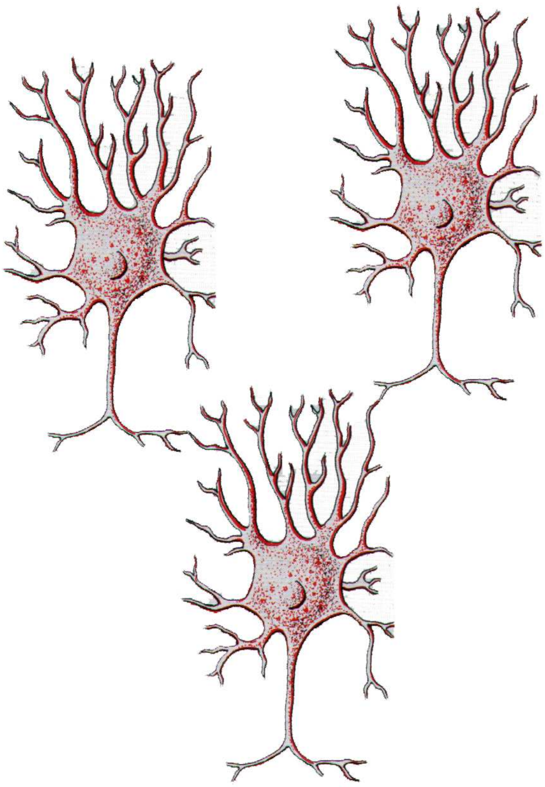
Neuron, consisting of

dendrites, soma, and axon.

(Purkinje cell from the cerebellum.)

Photography: Spektrum der Wissenschaft 10, 2001

## Biological Neural Networks



Potential is propagated from dendrite to soma.

If accumulated potential exceeds a threshold, the neuron fires.

Then potential is propagated along the axon to the next neurons.

Picture: Birbaumer & Schmidt, Biologische Psychologie, Springer, $^{2}$1991

## Artificial Neural Networks

Mathematical model, abstracts from certain properties.

Neurons hold real numbers.

Propagation needs no time.

All neurons are synchronized in discrete time steps.

Potentials accumulate as weighted sums.

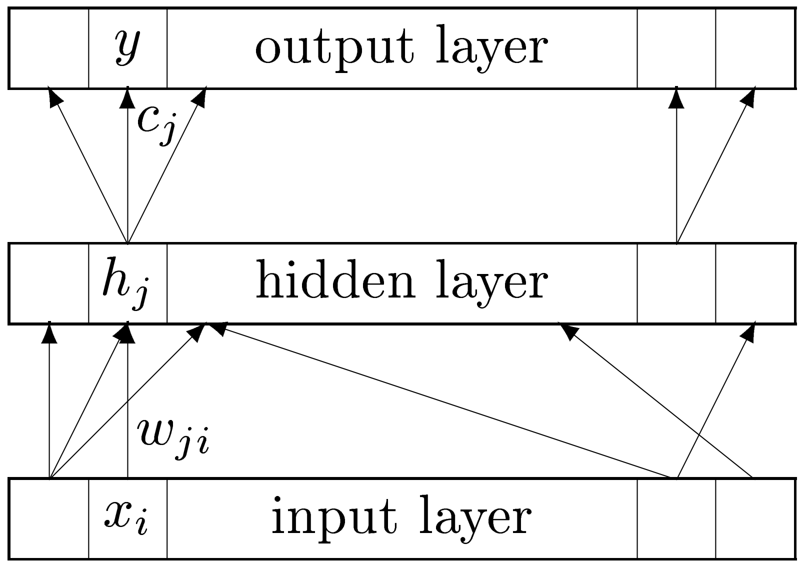
Threshold is differentiable.

Other models exist. The study of these abstract models is sometimes called *connectionism*.

## Artificial Neural Networks

A *3-layer feedforward network* (3ffn) consists of

- finitely many computational units
- organized in three layers:
  - input layer, hidden layer, output layer
- weighted connections between units
  - from input to hidden layer and
  - from hidden to output layer.




| | |
|---|---|
| $x_i$ | inputs |
| $w_{ji}$, $c_j$ | connection weights |
| $y$ | output |

## Artificial Neural Networks

The input-output function $f : \mathbb{R}^n \to \mathbb{R}$ is

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

with *thresholds* $\theta_j \in \mathbb{R}$ and *squashing function* $\phi$.

$\phi : \mathbb{R} \to \mathbb{R}$ is the same for each unit and usually

- non-constant, bounded, monotonic increasing,
- sometimes continuous.

Other architectures exist.

Neural networks can be trained from examples!

## Some Previous Work

Propositional Logic

$$(A \wedge (A \rightarrow B)) \rightarrow B$$

McCulloch and Pitts 1943
Pinkas 1991
Towell and Shavlik 1994 (KBANN)

Reflexive reasoning.
Shastri and Ajjanagadde 199x (SHRUTI)

Inference for Horn logic.
Hölldobler 1993 (CHCL)

Hybrid neuro-fuzzy systems.

etc.

Propositional case is fairly well understood.

## Representing Variabe Bindings

Datalog

$$\text{grandfather}(x, y) \leftarrow \text{father}(x, z) \wedge \text{parent}(z, y)$$

How to represent Datalog such that variable bindings can be preserved/evaluated?

Ashish Darbari 2000 (Dresden, Master thesis):

- Allow firing times (spiking neurons).
- Simultaneous firing represents variable binding.
- Combine with Hebbian learning.

We are currently just starting to investigate this in depth.

## Representing First Order Logic

$\text{even}(0) \quad \wedge$
$(\text{even}(s(x)) \leftarrow \neg\text{even}(x))$

How to represent “infinitely” many objects with a finite network?

Idea (Hölldober, Störr and Kalinke 1999):

- Restrict to (normal) logic programs.
- Represent program by a function.
- Encode function as function on $[0, 1]$.
- Approximate this function by neural networks.

Approach was extended by Hitzler and Seda 2000 & 2001.

## Logic Programs

(E.g. Prolog.)

A logic program $P$ is a finite set of clauses

$$\forall(A \leftarrow L_1 \wedge \cdots \wedge L_n)$$

from first order logic, usually written as

$$A \leftarrow L_1, \ldots, L_n,$$

where $A$ an atom, $L_i$ a literal, $n \geq 0$.
$B_P$: Herbrand base (all ground atoms).
$I_P = 2^{B_P}$: set of all Herbrand interpretations.
ground($P$): set of all ground instances
of clauses of $P$.

Define (non-monotonic) operator $T_P : I_P \to I_P$ by
$T_P(I)$ is set of all $A \in B_P$
for which there is a clause $A \leftarrow L_1 \wedge \cdots \wedge L_n$
in ground($P$) s.t. $I \models L_1 \wedge \cdots \wedge L_n$.

$I$ is a *supported model* iff $T_P(I) = I$.

$T_P$ basically encodes $P$.

# Logic Program Example

Program $P$:

$$p(0) \leftarrow$$
$$p(s(X)) \leftarrow \neg p(X)$$

$B_P = \{p\left(s^n(0)\right) : n \in \mathbb{N}\}$

$I_P = 2^{B_P}$

$$\text{ground}(P) = \Big\{p(0) \leftarrow \quad \Big\}$$
$$\bigcup \quad \Big\{p\left(s^{n+1}(0)\right) \leftarrow p\left(s^n(0)\right) \quad : n \in \mathbb{N}\Big\}$$

$$T_P\Big( \quad \{p(s(s(0))\} \quad \Big) = B_P \setminus \{ \quad p(s(s(s(0)))) \quad \}$$

$$T_P(M) = M \quad \Longleftrightarrow \quad M = \left\{p\left(s^{2n}(0)\right) \quad : n \in \mathbb{N}\right\}$$

## LPs versus ANNs

Neural Networks:

- approximates ("interpolates") functions
- hardly any knowledge about the fct$^{n}$ needed
- trained using incomplete data
- declarative semantics not available
- recursive networks hardly understood
- symbolic data difficult to represent

Logic Programming:

- direct implementation of relations
- explicit expert knowledge required
- highly recursive structure
- well understood declarative semantics
- symbolic data easy to represent

Seek the best of both paradigms!

## Functions via Neural Networks

**Theorem** (Funahashi 1989, simplified version):

$\phi$ non-constant, bounded, monotone increasing, continuous.
$K \subseteq \mathbb{R}$ compact,
$f : K \to \mathbb{R}$ continuous,
$\varepsilon > 0$.

Then exists a 3ffn with squashing fct$^{\mathrm{n}}$ $\phi$ and input-output function $\bar{f} : K \to \mathbb{R}$ with

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

$d$ metric which induces natural top. on $\mathbb{R}$.

- "Each continuous function $f : K \to \mathbb{R}$ can be uniformly approximated by input-output functions of 3ffns."

For example: $f : [0,1] \to [0,1]$.

## Structure-preserving mappings

$T_P : I_P \to I_P$.

$f : [0,1] \to [0,1]$.

We want to understand $T_P$ as a continuous function on $[0,1]$.

Seek $\iota : I_P \to [0,1]$
which is "structure preserving".

Mathematically: A (iso)morphism.
Homeomorphisms preserve "continuity structure".

Exists $\iota : I_P \to \mathcal{C}$.
$\mathcal{C}$ *Cantor set*, fractal (known from chaos theory).
$\mathcal{C} \subseteq [0,1]$.

From Tietze's Lemma: If $T_P$ is continuous on $I_P$, then $\iota(T_P)$ can be extended to a continuous function on $[0,1]$.

Then we can approximate with neural network.

## Continuity of $T_P$

"Continuity structure" on $I_P$:
called a *topology*.

Here: atomic topology $Q$.

Seda 1995,
building on Batharek and Subrahmanian 1989.

Further studied by Hitzler and Seda 199x.

Hitzler 2001 (PhD thesis).

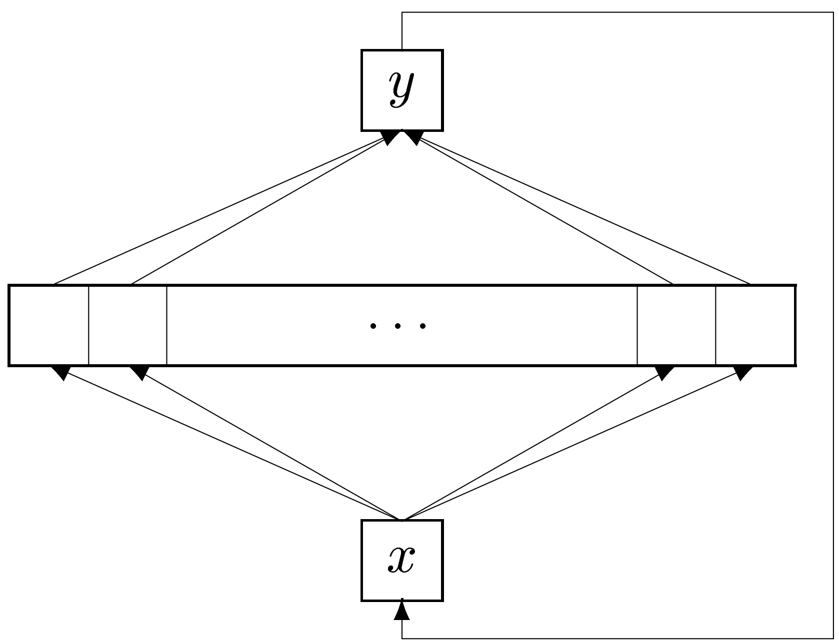
Amongst other things: Logical characterization for continuity of $T_P$.

Studied: generalizations, denotational (fixed-point) semantics.

## Recurrent Architecture I

- Approximation results give no way of constructing the network.
- Is the obtained approximation sufficient?

[HKS 1999] use the following recurrent architecture:




- Network iterates $T_P$.

## Recurrent Architecture II

$T = T_P$ consequence operator.
$f$ I/O function of approximating network.

For any $I \in I_P$ and any $n \in \mathbb{N}$ we have

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

$\lambda$ Lipschitz const. of extension of $\iota(T)$ onto $[0,1]$.
$\varepsilon$ error of the network.

## Conclusions

Much work remains to be done.

Continue the study of variable bindings.

What is the right connectionist paradigm?

We will continue the work in Dresden.

We seek students for our

**International Master Programme in Computational Logic.**

(English language, two years duration, Bachelor required.)

*Acknowledgements*: New results joint work with A.K. Seda. Thanks for discussions on the subject to H. Blair, S. Hölldobler, H.-P. Störr