

# A Note on the Relationships between Logic Programs and Neural Networks

Pascal Hitzler and Anthony Seda

IWFM2000, Maynooth, July 2000

## Contents

We want to represent operators associated with Logic Programs by Artificial Neural Networks. We build on an approach by Hölldobler, Kalinke and Störr [HKS] 199x.

- Logic Programs
- Feedforward Networks
- Preliminary Results
- Representing Continuous Operators
- Representing General Operators
- Discussion

Dept. of Mathematics, University College, Cork, Ireland  
{phitzler,aks}@ucc.ie <http://maths.ucc.ie/~{pascal,aks}/>

## Logic Programs

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

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

where  $A$  an atom,  $L_i$  a literal,  $n \geq 0$ .

$B_P$ : Herbrand base.

$I_P = 2^{B_P}$ : set of all Herbrand interpretations.

$\text{ground}(P)$ : set of all ground clauses of  $P$ .

Define (nonmonotonic) operator  $T_P : I_P \rightarrow I_P$  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  $\text{ground}(P)$  s.t.  $I \models L_1 \wedge \cdots \wedge L_n$ .

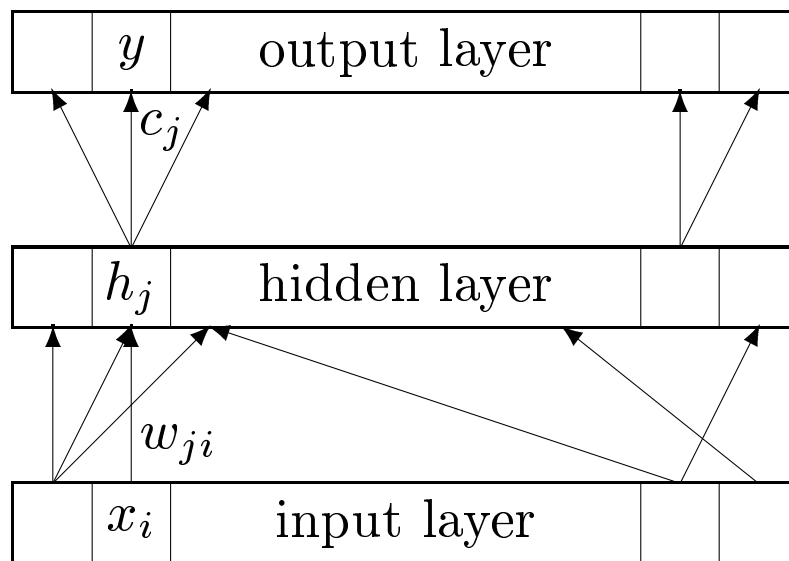
$I$  is a *supported model* iff  $T_P(I) = I$ .

$P$  can essentially be represented by  $T_P$ .

# Artificial Neural Networks I

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 II

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

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

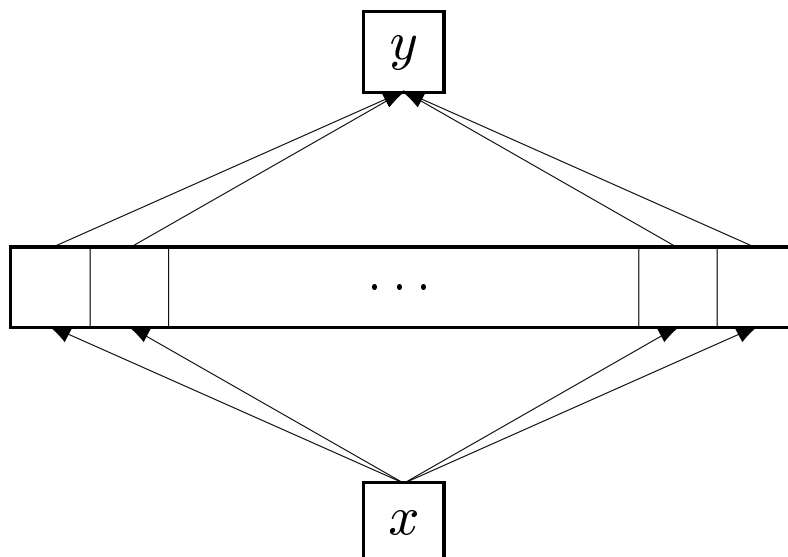
with *thresholds*  $\theta_j \in \mathbb{R}$  and *squashing function*  $\phi$ .

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

- nonconstant, bounded, monotonic increasing,
- sometimes continuous.

The following architecture will suffice:

- one input unit  $x$
- one output unit  $y$



## LPs versus ANNs

### Neural Networks:

- approximates (“interpolates”) functions
- hardly any knowledge about the  $fct^n$  needed
- trained using incomplete data
- successful in practical applications
- 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!

## The Cantor Topology

$B_P$  countable.

$I_P$  power set of  $B_P$ .

Identify with  $2^{B_P}$ , functions from  $B_P$  to  $\{0, 2\}$ .

\* Identify  $I_P$  with *Cantor set*  $\mathcal{C} \subseteq [0, 1] \subseteq \mathbb{R}$ ,  
 $\mathcal{C}$  real numbers in  $[0, 1]$  with ternary  
expansion using only digits 0 and 2.

$\mathcal{C}$  with subspace topology from  $\mathbb{R}$ : *Cantor space*.

Equivalent: product topology of discrete  $\{0, 2\}$ .

Corresponding topology on  $I_P$ : *atomic top.*  $Q$ .

Subbase:

$$\mathcal{G} = \{\mathcal{G}(A) : A \in B_P\} \cup \{\mathcal{G}(\neg A) : A \in B_P\},$$

$$\mathcal{G}(L) = \{I \in I_P : I \models L\}, \quad L \text{ literal.}$$

$(I_P, Q)$ ,  $\mathcal{C}$  homeomorphic.

$\iota : I_P \rightarrow \mathcal{C}$  homeomorphism  
(uncountably many exist).

$\mathcal{C}$  compact subset of  $\mathbb{R}$ .

## Approximating Continuous $T_P$ I

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

$\phi$  nonconstant, bounded, monotone increasing,  
continuous.

$K \subseteq \mathbb{R}$  compact,

$f : K \rightarrow \mathbb{R}$  continuous,

$\varepsilon > 0$ .

Then exists a 3ffn with squashing fct<sup>n</sup>  $\phi$  and  
input-output function  $\bar{f} : K \rightarrow \mathbb{R}$  with

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

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

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

$T_P : I_P \rightarrow I_P$  continuous, then

$\iota(T_P) : \mathcal{C} \rightarrow \mathcal{C} : x \mapsto \iota(T_P(\iota^{-1}(x)))$  continuous.

## Approximating Continuous $T_P$ II

**Theorem** (Seda 1995)

$P$  normal logic program.

$T_P$  continuous in  $Q$  iff

for each  $I \in I_P$ ,  $A \in B_P \setminus T_P(I)$  either

exist no clause with head  $A$  or exist finite set

$S(I, A) = \{A_1, \dots, A_k, B_1, \dots, B_{k'}\} \subseteq B_P$  with

(i)  $A_1, \dots, A_k \in I$  and  $B_1, \dots, B_{k'} \notin I$ ,

(ii) in any clause  $C$  with head  $A$ ,

at least one  $\neg A_i$  or at least one  $B_j$

occurs in the body of  $C$ .

**Theorem**

The  $T_P$ -operator for  $P$  satisfying the conditions of the theorem above can be uniformly approximated by input-output mappings of 3ffns.

This holds for example for  $\Phi_\omega$ -accessible programs with injective level mapping.

(Hitzler & Seda 2000)

## Approximating General $T_P$ I

**Theorem** (Hornik, Stinchcombe, White 1989, simplified version)

$\phi : \mathbb{R} \rightarrow (0, 1)$  monotone increasing, surjective.

$f : \mathbb{R} \rightarrow \mathbb{R}$  Borel-measurable,

$\mu$  probability Borel-measure on  $\mathbb{R}$ ,

$\varepsilon > 0$ .

Then exists 3ffn with squashing fct<sup>n</sup>  $\phi$  and input-output function  $\bar{f} : \mathbb{R} \rightarrow \mathbb{R}$  with

$\varrho_\mu(f, \bar{f}) =$

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

- “The class of functions computed by 3ffns is dense in the set of all Borel measurable functions  $f : \mathbb{R} \rightarrow \mathbb{R}$  rel. to the metric  $\varrho_\mu$ .”

**Theorem**

$P$  normal logic program:

$T_P$  is measurable on  $(I_P, \sigma(\mathcal{G})) = (I_P, \sigma(Q))$

and  $\iota(T_P)$  can be approximated

in the sense of the theorem above.

## Approximating General $T_P$ II

Approximation is only *almost everywhere*  
i.e. except a set of measure 0.

But two degrees of freedom:

- Choice of homeomorphism  $\iota$ .
- Expansion of  $\iota(T_P)$  onto all of  $[0, 1]$ .

### Theorem

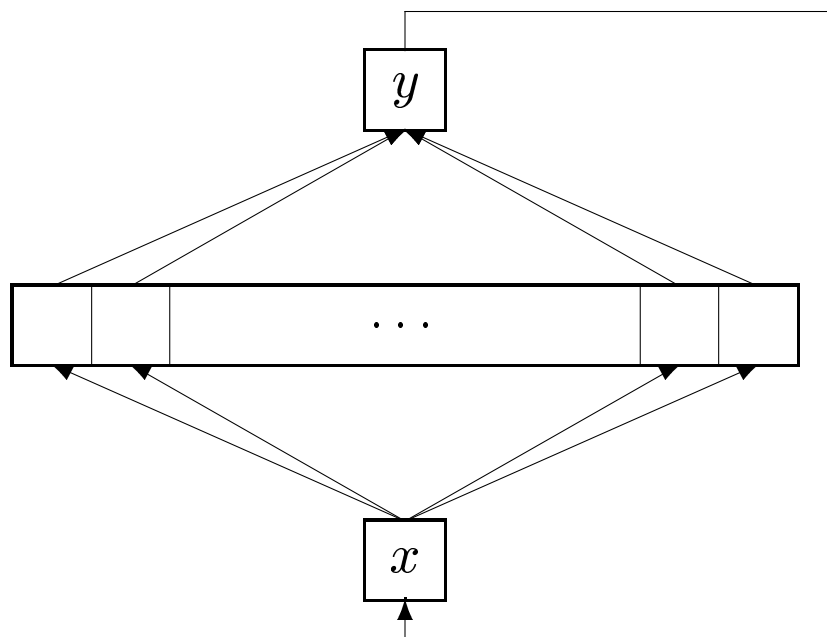
$\iota$  and expansion can be chosen such that  
for each  $I \in I_P$  which is finite or cofinite  
there exists  $x \in [0, 1]$  with  $|x - \iota(I)| < \varepsilon$  and  
 $|\iota(T_P(I)) - f(x)| < \varepsilon$ .

- \* “piecewise linear interpolation” of  $T_P$ .
- \*  $\iota$  obtained by “ternary expansion”.

## Discussion/Further Work I

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

[HKS] use the following recurrent architecture:



- Network iterates  $T_P$ .
- $P$  acyclic:  $\iota$  can be found such that network settles down into unique stable state corresponding to declarative semantics of  $P$ .
- Carries over to  $\Phi_\omega$ -accessible programs?

## Discussion/Further Work II

- Use symbolic representation of infinite  $B_P$  to circumvent the approximation? [HKS]
- Use propositional quantitative logic programming in order to give declarative semantics to (recurrent) networks?
  - \* Probably change of network architecture needed.
  - \* May give rise to a new quantitative logic programming paradigm.
- Yan Zhang 1999 uses architecture of hybrid symbolic-neural networks and obtains a relationship between them and the answer set semantics in propositional logic programming.