Neural-Symbolic Integration
A self-contained introduction
Sebastian Bader  Pascal Hitzler
ICCL, Technische Universität Dresden, Germany
AIFB, Universität Karlsruhe, Germany

Outline of the Course

▶ Introduction and Motivation
▶ The History of Neural-Symbolic Integration
▶ The Core Method for Propositional Logic
▶ The Core Method for First-Order Logic

Why Neural Symbolic Integration

As we will see, connectionist systems and symbolic AI systems have quite contrasting advantages and disadvantages. We try to integrate both paradigms while keeping the advantages.

The Neural Symbolic Cycle

Connectionist Systems

▶ Inspired by nature.
▶ Massively parallel computational model.
▶ A Connectionist System consist of ...
  ▶ a set \( U \) of units (input, hidden and output).
  ▶ a set of connections \( C \subseteq U \times U \), each labelled with a weight \( w \in \mathbb{R} \).

Units of Connectionist Systems

A unit is characterised by ...
▶ Activation function, mapping inputs \( i \) to the potential \( p \):

\[
p = \sum_{n} i_{n} \cdot w_{n}
\]

▶ Output function, mapping the potential \( p \) to the output \( o \):

threshold  ramp  sigmoidal  tanh  Gaussian

Dynamics of a Network

Input: \( p \) set from outside

Hidden output:

Activation F.:

\[
p = \sum_{n}(i_{n} - w_{n})^{2}
\]

Output F.:

\[
o = p - \sigma^{2}
\]

Activation F.:

\[
p = \sum_{n}(i_{n} \times w_{n})
\]

Output F.:

\[
o = p
\]
Network learns the map between letters and phonemes

Idea: minimise

Terrence J. Sejnowski and Charles R. Rosenberg, 1987

Ken-Ichi Funahashi publishes

Let a set of samples

Trained using samples of the form:

Learning as generalization.

Compute the part of the error caused by the hidden units.

Neural-Symbolic Integration

Connectionist Systems

Symbolic AI

Neural-Symbolic Integration

Backpropagation

Let a set of samples \( \{(i_1, o_1), \ldots, (i_n, o_n)\} \) be given.

Error of the network: \( E = \sum (\mathcal{N}(i) - o)^2 \).

Idea: minimise \( E \) by gradient descent.

A sample run ...

Prior training:

After training:

Funahashi’s Theorem

Theorem (Ken-Ichi Funahashi, 1989)

Every continuous function \( f : K \to \mathbb{R} \) (with \( K \subset \mathbb{R}^n \) compact) can be approximated arbitrarily well using 3 layer feed-forward networks with sigmoidal units.

History of Connectionist Systems

1943 Warren Stigl McCulloch and Walter Pitts publish

“A logical calculus of the ideas immanent in nervous activity”.

1968 Marvin Minsky and Seymour Papert publish

“Perceptron”.


1989 Ken-Ichi Funahashi publishes

“On the Approximate Realisation of Continuous Mappings by Neural Networks”.

NETtalk

Terrence J. Sejnowski and Charles R. Rosenberg, 1987

Network learns the map between letters and phonemes

3-layer feed-forward network with sigmoidal units:

- 203 input units: encoding a window of 7 letters
- 80 hidden units
- 26 output units: representing phonemes, punctuation ...

Trained using samples of the form:

Word phonemes stress and syllable

logic I\(\text{AJK} \)

programme I\(\text{grm@m-} \)

neural \( \text{nu-r-L} \)

network \( \text{nEt-r-Rk} \)
We play Backgammon symbolically.

ALVINN (Autonomous Land Vehicle In a Neural Network),
- Pomerleau 1993
- Learns to control NAVLAB vehicles by watching a person.
- 3-layer feed-forward network with sigmoidal units:
  - 960 input units: 30x32 units serve as two dimensional retina
  - 5 hidden units
  - 30 output units: representing the steering direction

MANIAC (Multiple ALVINN Networks In Autonomous Control)
- Jochem et al 1993
- Multiple ALVINN networks, each for a certain type of road.

ALVINN & MANIAC

The road for ALVINN, MANIAC & RALPH:

RALPH (Rapidly Adapting Lateral Position Handler)
- Pomerleau 1995
- Drove in 9 days from Pittsburgh to San Diego (2850 but 50 miles)

TD-Gammon

Artificial Backgammon player (Gerry Tesauro, 1995):
- Based on standard neural network.
- Learns by playing against itself.
- Reaches championship level.

Btw.:
- We play Backgammon symbolically.
- Can we learn from the network to play better?

Symbolic AI

Inspired by human thinking.
- Aristotle tried to formalize human thinking by means of logic (500BC).
- First “is-a” hierarchy by Porphyry of Tyros (280).
- Use of logic-like languages for knowledge representation and reasoning.

SHRDLU

Terry Winograd, MIT, 1968-70
- SHRDLU carried on a simple dialog about a small world.
- A sample dialog:
  - Person: Pick up a red block.
  - Computer: Ok. (does it)
  - Person: Grasp the pyramid.
  - Computer: I don't understand which pyramid you mean.
  - Person: Find a block which is taller than the one you are holding and put it into the box.
  - Computer: By 'it', I assume you mean the block which is taller than the one i am holding.
  - Computer: Ok. (does it)
  - Person: What does the box contain?
  - Computer: The blue pyramid and the blue block.

SHRDLU - Internals

- Written in LISP.
- Equipped with some knowledge about the world:
  (!(!IS :B1 #BLOCK))
  (!(!IS :RED #COLOR))
  (!(!IS :B2 #PYRAMID))
  (!(!IS #GREEN #COLOR))
  (!(!IS :B3 #BLOCK))
  (!(!IS #BLACK #COLOR))
  (!(!COLOR :B1 :RED))
  (!(!COLOR :B2 #GREEN))
  (!(!SHAPE :B1 #RECTANGULAR))
  (!(!COLOR :TABLE #BLACK))
  (!(!SHAPE :B3 #RECTANGULAR))
  (DEFPROP TA-AT (THANTE (X Y) (#AT $X $Y))
   (THREPLACA (CDR (ATAB $?X)) $?Y))
  (THRSUCCEED)

- Can be downloaded from
  http://hci.stanford.edu/winograd/shrdlu

ProLog (Programming In Logic)

- Designed as a tool for man-machine communication in natural language.
- Philippe Rousseld and Alain Colmerauer, 1972
- The first Prolog-Application:
  Every psychiatrist is a person.
  Every person he analyzes is sick.
  Jacques is a psychiatrist in Marseille.
  Is Jacques a person? Yes.
  Where is Jacques? In Marseille.
  Is Jacques sick? I don’t know.

- Consisted of 610 clauses.
Applications Involving Prolog

Nowadays:
- Turing complete programming language.
- Usually with additional (non-logical) features.

Some application areas:
- Expert and rule systems.
- Computational linguistics (e.g. representation of grammars).
- Planning in AI.
- Cognitive robotics.
- Semantic web.

Deterministic Finite Automata

A Moore Machine consists of:
- $Q$ - set of states with an initial state $q_0 \in Q$
- $\Sigma$ - set of input symbols
- $\Delta$ - set of output symbols
- $\delta$ - state transition function $\delta : Q \times \Sigma \rightarrow Q$
- $\lambda$ - state output function $\lambda : Q \rightarrow \Delta$

Example

$$Q = \{q_0, q_1\}$$
$$\Sigma = \{a, b\}$$
$$\Delta = \{(0, 1)\}$$
$$\delta = \{(q_0, a, q_0), (q_1, b, q_1)\}$$
$$\lambda = \{(q_0 \mapsto 1, q_1 \mapsto 0)\}$$

Properties of Symbolic Systems

- Human readable and writable, i.e. background knowledge is directly integrable.
- Declarative semantics is available.
- Recursive structures can easily be represented and manipulated.
- Successfully used in many application areas.
- Hard to learn and to adapt to new environments.
- If parts of the system breaks, the whole system fails.
- Reasoning can be very hard.

Why Neural-Symbolic Integration?

- Connectionist systems and symbolic knowledge representation are two major approaches in AI.
- Both have complementary advantages and disadvantages.
- We try to integrate both by keeping the advantages:
  - Human readable and writable.
  - Declarative semantics is available.
  - Recursive structures can easily be represented and manipulated.
  - Massively parallel paradigm.
  - Well suited to learn and to adapt to new environments.
  - Gracefully degradation.

Major Problems in Neural-Symbolic Integration

- How can symbolic knowledge be represented within connectionist systems?
- How can symbolic knowledge be extracted from connectionist systems?
- How can symbolic knowledge be learned using connectionist systems?
- How can connectionist learning be guided by symbolic background knowledge?
A McCulloch-Pitts network consist of ...  
- A set $I$ of input units.
- A set $U$ of binary threshold units.
- A subset $O \subseteq U$ of output units.

**Example**

$$I = \{x, y\}$$
$$U = \{h, a\}$$
$$O = \{a\}$$

**McGrath-Pitts Networks**

**From Moore Machines to McCulloch-Pitts Networks**

**Conclusions**

- McCulloch-Pitts networks are finite automata and vice versa.
- The paper ("A logical calculus of the ideas immanent in nervous activity") started the research on artificial neural networks and on finite automata.

- Similar constructions work for other types of automata.
Recursive Autoassociative Memory (RAAM)

- Designed to encode structured data, e.g., trees:

- Terminals are mapped to vectors:
  - $A \rightarrow (1,0,0,0)$
  - $B \rightarrow (0,1,0,0)$
  - $C \rightarrow (0,0,1,0)$
  - $D \rightarrow (0,0,0,1)$

- Nonterminals are learned.

RAAM - Conclusions

- For details, see (Pollack, 199x).
- Efficiently implementable.
- Use of powerful gradient based learning techniques.
- System degrades gracefully.
- Difficulties to distinguish terminals and non-terminals for terms with depth $\geq 5$.
- Capacity limit $\approx$ depth 5.
- System needs an external controller.
- Demonstration that structured data can be represented within a connectionist system.

SHRUTI - A Sample Knowledge Base

- Rules:
  - $\text{owns}(Y,Z) \rightarrow \text{gives}(X,Y,Z)$
  - $\text{owns}(X,Y) \rightarrow \text{buys}(X,Y)$
  - $\text{can-sell}(x,y) \rightarrow \text{owns}(X,Y)$

- Facts:
  - $\text{gives}(\text{john}, \text{josephine}, \text{book})$
  - $(\exists X)b\text{buys}(\text{john}, X)$
  - $\text{owns}(\text{josephine}, \text{book})$

- Question:
  - $\text{can-sell}(\text{josephine}, \text{book})$? yes
  - $(\exists X)\text{owns}(\text{josephine}, X)$? yes ($X \rightarrow \text{book}, X \rightarrow \text{ball}$)

SHRUTI - A Sample Network Run

SHRUTI - Conclusions

- Answers are derived in a time proportional to the depth of the search space (Reflexive Reasoning)
- Network size is linear in the size of the knowledge base.
- A rule can be used only a fixed number of times.
- Biologically plausible.
The threshold of conjunctive units are computed as follows:

\[ \theta = (P - 0.5) \cdot w \]

- Support of negation and inconsistency (Shastry & Wendelken, 1999).
- Multiple instantiation of a single rule (Wendelken & Shastry, 2004).

Simple Learning using Hebbian Learning
Refinement using standard backpropagation.
Successfully applied to a number of problems
Multiple instantiation of a single rule
Mapping of hierarchical domain knowledge into a connectionist system.
Support of negation and inconsistency (Shastry & Wendelken, 1999).

Can simple “if-then” rules be represented and learned using a connectionist architecture?
- Geoffrey G. Towell and Jude W. Shavlik, 1994

For any value of \( w \) we can compute an \( n \), such that the \( o_C \) exceeds any threshold to be considered active.
- Can be solved using bipolar output functions.

Works well if rules have only few conditions and there are only few rules with the same consequence, but:
- Towell and Shavlik used sigmoidal output functions:

\[ o = \frac{1}{1 + e^{-(p-n)}} \]

- The threshold of conjunctive units are computed as follows:

\[ \theta = (P - 0.5) \cdot w \]

Where \( P \) is the number of positive antecedents.
- The threshold of disjunctive units is always set to

\[ \theta = w/2 \]

Mapping of hierarchical domain knowledge into a connectionist system.
Refinement using standard backpropagation.
Successfully applied to a number of problems (e.g. DNA sequence analysis).
Outperforms purely empirical and purely hand-built classifiers.
Symmetric Networks and Logic Formulae

- It is possible to associate an energy function $E(t)$ describing the state of the network at time $t$.
- The energy is monotone decreasing, i.e. $E(t) \geq E(t + 1)$.
- Is there a link between propositional logic formulae and symmetric networks (Pinkas, 1991)?
- To each propositional logic formula we can define a function $\tau$ which is "compatible" with the error function.
- We can construct a symmetric network such that the activation of the network at the minima coincide with the models of the formula.

Symmetric Networks - A Simple Example

$F = (\neg o \lor m) \land (\neg s \lor \neg m) \land (\neg c \lor v) \land (\neg v \lor \neg m)$

\[ \tau(F) = vm - cm - cs + sm - om + 2c + o \]

Symmetric Networks - Conclusions

- Strong link between propositional logic formulae and symmetric networks.
- Further extensions to non-monotonic logics and inconsistency.
- Add penalties to clauses which define a preference.
- Network settles down to most preferable interpretation.

Symmetric Networks and Logic Formulae

Relate logic programs and connectionist systems

Embed interpretations into (vectors of) real numbers.

Hence, obtain an embedded version of the $T_P$-operator.

Construct a network computing one application of $f_P$.

Add recurrent connections from output to input layer.

The Core Method

- How can symbolic knowledge be represented within connectionist systems? (What is $\tau$?)
- How can symbolic knowledge be extracted from connectionist systems? (What is $\tau^{-1}$?)
- How can symbolic knowledge be learned using connectionist systems?
- How can connectionist learning be guided by symbolic background knowledge?

Neural-Symbolic Integration

A self-contained introduction

Sebastian Bader Pascal Hitzler

ICCL, Technische Universität Dresden, Germany

AIFB, Universität Karlsruhe, Germany

Part

The Core Method for Propositional Logic
Propositional Logic Programs – An Example

\[ \begin{align*}
A &\iff \neg B. & \text{\% } A \text{ is true, if } B \text{ is false.} \\
B &\iff A \land \neg B. & \text{\% } B \text{ is true, if } A \text{ is true and } B \text{ is false.} \\
B &\iff B. & \text{\% } B \text{ is true, if } B \text{ is true.}
\end{align*} \]

Propositional Logic Programs – The Syntax

Definition (Propositional Variables & Connectives)
\[ A, B, C, D, \ldots \land = "and" \quad \iff = "if-then" \quad \neg = "not" \]

Definition (Clause)
\[ H \iff L_1 \land L_2 \land \ldots \land L_n, \quad \text{head with } L_i \text{ either } X \text{ or } \neg X \]

Definition (Propositional Logic Program)
A propositional logic program is a finite set of clauses.

Propositional Logic Programs – The Semantics Ctd.

Example (Models of the running example)
\[ \begin{align*}
\{A\} &\models P \\
\{B\} &\not\models P \\
\{A, B\} &\models P
\end{align*} \]

Propositional Logic Programs – The Semantics

Definition (Herbrand Base \(B_C\))
The Herbrand base is the set of all variables occurring in \(P\).

Example (\(B_C\) for the running example)
\[ B_C = \{A, B\} \]

Definition (Interpretation)
An interpretation is a subset of the Herbrand base.

Example (Interpretations for the running example)
\[ I_1 = \emptyset \\
I_2 = \{A\} \\
I_3 = \{B\} \\
I_4 = \{A, B\} \]

Propositional Logic Programs – The Semantics Ctd.

Definition (Model)
An interpretation \(M\) satisfying every clause of a program \(P\) is called a model of \(P\) (in symbols \(M \models P\)).

Example (Models of the running example)
\[ \begin{align*}
A &\iff \neg B. & 0 \not\models P \\
B &\iff A \land \neg B. & (A) \not\models P \\
B &\iff B. & (B) \models P \\
(A, B) &\models P
\end{align*} \]

Constructing the Core-Network

1. For each element of \(B_C\), add an input unit and an output unit with threshold 0.5.
2. For each clause \(H \iff L_1 \ldots L_n\), do the following:
   2.1 Add a hidden unit \(c\) and a connection to \(H\) \((w = 1.0)\).
   2.2 Connect every \(L_i\) and \(c\) with \(w = \begin{cases} 1.0 \text{ if } L_i \text{ is positive,} \\ -1.0 \text{ if } L_i \text{ is negated.} \end{cases}\)
   2.3 Set the threshold of \(c\) to \("number of pos. \cdot 0.5"\)

Example
\[ \begin{align*}
A &\iff \neg B. \\
B &\iff A \land \neg B. \\
B &\iff B.
\end{align*} \]

One Application of \(T_P\)
\[ \begin{align*}
\{\} &\models (A) \\
\{A\} &\models (A, B) \\
\{B\} &\models (B) \\
\{A, B\} &\models (B)
\end{align*} \]
Retteve Application of $T_P$

$$
A \leftarrow \neg B.
B \leftarrow A \land \neg B.
B \leftarrow B.
$$

Main Results (Hölldobler & Kalinke, 1994)

- 2-layer networks cannot compute $T_P$.
- For each program $P$ there exists a 3-layer kernel computing $T_P$.

Space and Time Complexity

Let $n$ be the number of clauses, $m$ be the number of propositional variables:

- $2m + n$ units, $2mn$ connections in the kernel.
- $T_P(l)$ is computed in 2 steps.
- The parallel model to compute $T_P$ is optimal.
- The recurrent network settles down in at most $3n$ steps.

Extraction - A Pedagogical Approach

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$A'$</th>
<th>$B'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1.5</td>
<td>1.5</td>
<td>3.0</td>
<td>1.8</td>
<td>1.2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.0</td>
<td>2.0</td>
<td>0.0</td>
<td>0.6</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2.5</td>
<td>1.7</td>
<td>1.3</td>
<td>0.6</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Extraction Methods

- Single units do not necessarily correspond to single rules.
- In general: It is NP-complete to find the minimal logical description for a trained network (Golea, 1996).
- There is not always a single minimal program (Lehmann, Bader & Hitzler, 2005).

Declarative

Pedagogical

Extraction - A Pedagogical Approach

\[
\begin{align*}
A & \leftarrow \neg A \land \neg B. \\
A & \leftarrow \neg A \land B. \\
A & \leftarrow A \land \neg B. \\
A & \leftarrow A \land B. \\
B & \leftarrow \neg A \land B. \\
B & \leftarrow A \land \neg B. \\
B & \leftarrow A \land B.
\end{align*}
\]

Sound, i.e. every extracted rule is a rule implemented by the network.

Complete, i.e. every rule implemented by the network will be extracted.

Bad time-complexity, due to the exponential blow-up.

Does not create the smallest program automatically.
Extraction – A Decompositional Approach

We can do much better (Mayer-Eicherberger, 2006):

- Decompositional approach.
- Implementable (the implementation is under way).
- Sound.
- Complete.
- Create very small programs automatically.

The CILLP-System

? Can the learning capabilities of KBANN be combined with the Core Method (Garcez & Zaverucha, 1999)?
- Using sigmoidal functions, we obtain a standard 3-layer feed-forward neural network.

This network is trainable using back-propagation.

CILLP - Extracting a Learned Program

- The pedagogical approach would work, but ...
- The decompositional approach mentioned above does not work for sigmoidal units.
- Garcez, Broda & Gabbay (2001) proposed a suitable method, which ...
  - is sound.
  - is computational feasible due to clever restriction of the search space.
  - is not necessarily complete.
  - does not necessarily create the small programs.

CILLP - The MONK’s Problems

- Robots are described by 6 properties, e.g. head-shape ∈ {round, square, octagon}, ...
- Classification task: “Recognize robots with (body-shape = head-shape) or (jacket-color = red)”
- Network architecture:
  - 17 input units: one for each attribute.
  - 3 hidden layer units.
  - 1 output unit: indicating answer “yes” or “no”.
- 100% performance of the network and extracted rules.
- Pruning: from 131072 possible inputs for some hidden unit, only 18724 were queried.

CILLP - Conclusions

- Successfully used for ...
  - classification tasks like the MONK’s problem.
  - DNA sequence analysis (Promoter Recognition, Splice Junction Determination).
  - Power system fault diagnosis.
- Extensions of the CILLP-System:
  - Metalevel priorities between rules (Garcez, Broda & Gabbay, 2000).
  - Intuitionistic logic (Garcez, Lamb & Gabbay, 2003).
  - Modal logic (Garcez, Lamb, Broda & Gabbay, 2004).

Main Results (Hölldobler & Kalinke, 1994)

- 2-layer networks cannot compute $T_p$.
- For each program $P$ there exists a 3-layer kernel computing $T_p$.
- For each 3-layer kernel $K$ there exists a program $P$, such that $K$ computes $T_p$.
- Let $n$ be the number of clauses, $m$ be the number of propositional variables
  - $2n - n$ units, $2mn$ connections in the kernel.
  - $T(f)$ is computed in 2 steps.
  - The parallel model to compute $T_p$ is optimal.
- The recurrent network settles down in at most $3n$ steps.

The Core Method

- Relate logic programs and connectionist systems
  - Embed interpretations into (vectors of) real numbers.
  - Hence, obtain an embedded version of the $T_p$ operator.
  - Construct a network computing one application of $f_p$.
  - Add recurrent connections from output to input layer.
Major Problems in Neural-Symbolic Integration

- How can symbolic knowledge be represented within connectionist systems? (What is \( r^{-1} \)?)
- How can symbolic knowledge be extracted from connectionist systems? (What is \( r^{-1} \)?)
- How can symbolic knowledge be learned using connectionist systems?
- How can connectionist learning be guided by symbolic background knowledge?

_conclusions_

We have a complete system implementing the NeSy-Cycle for propositional logic programs.

![Neural-Symbolic Integration Diagram]

Neural-Symbolic Integration

A self-contained introduction

Sebastian Bader  Pascal Hitzler

ICCL, Technische Universität Dresden, Germany
AIFB, Universität Karlsruhe, Germany

First Order Logic Programs – Two Examples

- \( \text{nat}(0). \) % 0 is a natural number.
- \( \text{nat}(\text{succ}(X)) \leftarrow \text{nat}(X). \) % The successor \( \text{succ}(X) \) is a natural number if \( X \) is a natural number.
- \( \text{even}(0). \) % 0 is an even number.
- \( \text{even}(\text{succ}(X)) \leftarrow \text{odd}(X). \) % The successor of an odd \( X \) is even.
- \( \text{odd}(X) \leftarrow \neg \text{even}(X). \) % If \( X \) is not even then it is odd.

First Order Logic Programs – The Syntax

**Functions, Variables and Terms**

- \( \mathcal{F} = \{0, \text{succ}/1\} \)
- \( \mathcal{V} = \{X\} \)
- \( \mathcal{T} = \{0, \text{succ}(0), \text{succ}(X), \text{succ}(	ext{succ}(0)), \ldots\} \)

**Predicate Symbols and Atoms**

- \( \mathcal{P} = \{\text{even}/1, \text{odd}/1\} \)
- \( \mathcal{A} = \{\text{even}(\text{succ}(X)), \text{odd}(\text{succ}(0)), \text{odd}(0), \text{odd}(X), \ldots\} \)

First Order Logic Programs – The Semantics

**Herbrand Base** \( \mathcal{B}_c = \text{Set of ground atoms} \)

- \( \mathcal{B}_c = \{\text{even}(0), \text{even}(\text{succ}(0)), \ldots, \text{odd}(0), \text{odd}(\text{succ}(0)), \ldots\} \)

**Interpretations** = Subsets of the Herbrand base

- \( I_1 = \{\text{even}(\text{succ}^n(0)) \mid n \geq 1\} \)
- \( I_2 = \{\} \)
- \( I_3 = \{\text{odd}(\text{succ}^{n+1}(0)) \mid n \geq 0\} \)
- \( I_4 = I_2 \cup I_3 \)
**Bridging the Gap**  
*Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)*

**Some Results**

**Theorem (Hölldobler, Kalinke & Störr, 1999)**  
The $T_P$-operator associated with an acyclic (wrt. injective level mapping) first order logic program can be approximated arbitrarily well using standard sigmoidal networks.

Some conclusions and limitations:  
- The Core-Method can be applied to first order logic.  
- First treatment of first-order logic with function symbols in a connectionist setting.  
- No algorithm to construct the network.  
- Very limited class of logic programs.

**Problems**

- $B_L$ is usually infinite and therefore the propositional approach does not work.  
- How can we bridge the gap?  
  - How can first-order terms be represented?  
  - How can first-order rules be represented?  
  - How can the variable-binding be solved?

**Embedding First-Order Terms into the Real Numbers**

Using an injective level mapping, we can assign a unique real number to each interpretation:

$$i(I) = \sum_{A \subseteq I} 4^{-|A|}$$

This coincides with a "binary" representation:

$$B_L = \{ e(0), o(0), e(1), o(1), e(2), o(2), \ldots \}$$

$$i(\{e(0), o(1), e(2)\}) = 0.10010_4 = 0.25_{10}$$

$$i(\{e(0), e(1), e(2)\}) = 0.10101_4 = 0.27_{10}$$

**Constructions using sigmoidal and RBF-units are given in (Bader, Hitzler & Witzel, 2005).**
A Problem ...

- The accuracy of this approach is very limited.
- E.g., on a 32 bit computer, only 16 atoms can be represented.
- Therefore, we need to use real vectors instead of a single real number to represent interpretations.

Multi-dimensional Level Mappings

- A Multi-dimensional Level Mapping \( \| \cdot \| \) assigns to each ground atom \( a \in \mathbb{N}^+ \) and a dimension \( d \in \{1, \ldots, m\} \):

\[
\| e(s^n(0)) \| = (n + 1, 1) \quad \| o(s^n(0)) \| = (n + 1, 2)
\]

- ... still "enumerates" the Herbrand base:

Example (Even and odd numbers)

\[
\begin{array}{cccc}
\text{dim1} & e(0) & e(s(0)) & e(s(s(0))) \\
\text{dim2} & o(0) & o(s(0)) & o(s(s(0)))
\end{array}
\]

Embedding First-Order Terms into the Real Numbers

Using an injective \( m \)-dimensional level mapping, we can assign a unique \( m \)-dimensional vector to each interpretation:

\[
\iota(f) = \sum_{A \in f} \iota(A)
\]

\[
\iota(A) = \left( \iota_1(A), \ldots, \iota_m(A) \right)
\]

\[
\iota_i(A) = \begin{cases} 
4^{-i} & \text{for } |A| = (l, d) \text{ and } i = d \\
0 & \text{otherwise}
\end{cases}
\]

\( \iota \) - The Set of all embedded Interpretations

\( \iota_m \) for the 2-dimensional case:

\[
\iota_m = \{ \iota(f) | f \in \mathcal{L} \}
\]

Another construction:

\[
\iota(e(0)) \mapsto (0, 0.25, 0.25) \quad \iota(o(0)) \mapsto (0.25, 0.25)
\]

\[
\iota(e(0), o(0)) \mapsto (0.25, 0.25)
\]

Implementation

A first prototype implemented by Andreas Witzel (Witzel, 2006):

- Merging of the techniques described above and Supervised Growing Neural Gas (SGNG) (Fritzke, 1998).
- Radial basis function network approximating \( T_P \).
- Very robust with respect to noise and damage.
- Trainable using a version of backpropagation together with techniques from SGNG.
The sequence of attractors of interpolating IFSs for acyclic normal programs.

For a finite set of points taken from a FOL, there exists an IFS such that the attractor coincides with the graph of the embedded $T_P$-operator.

Let $P$ be a program such that $d_P$ is Lipschitz-continuous. Then there exists an IFS such that the attractor is the graph of $d_P$.

For a finite set of points taken from a $T_P$-operator, we can construct an interpolating IFS.

The sequence of attractors of interpolating IFSs for acyclic programs converges to the graph of the program.

IFSs can be encoded using RBF networks.

- Prototypical implementation.
- Very robust with respect to noise and damage.
- Trainable using more or less standard algorithms.
- System outperforms other architectures (at least for the tested examples).

- System requires many parameters.
- There is no first-order extraction technique yet.

First-order by propositional approximation

Let $P$ be definite and $I$ be its least Herbrand model (Seda & Lane, 2004):

- Choose some error $\varepsilon$.
- There exists a finite ground subprogram $P_n$ (least model $I_n$) such that $d(l, I_n) < \varepsilon$.
- Use propositional approach to encode $P_n$.
- Increasing $n$ yields better approximations of $T_P$.
  (If $T_P$ is continuous, $d$.)
- Approach works for other (many-valued) logics similarly.

Comparison of the approaches

- Seda & Lane:
  - For definite programs under continuity constraint.
  - Treatment of acyclic programs should be ok.
  - Better approximation increases all layers of network.
  - Step functions only.
  - Sigmoidal approach (learning) to be investigated.

- Bader, Hitzler & Witzel:
  - For acyclic normal programs.
  - Treatment of definite (continuous) programs should be ok.
  - Better approximation increases only hidden layer.
  - Variety of activation functions.
  - Standard learning possible.

Iterated Function Symbols

The Sierpinsky Triangle:
Extraction of First-Order Logic Programs

- Very little work has been done on this.
- A general idea:
  - Use any initialization method as a base.
  - Neural network are points in $\mathbb{R}^n$, where $n$ is number of weights.
  - Define conditions on programs which may be extracted (E.g.: maximum number of atoms or of term nesting depth).
  - ~ discrete points in $\mathbb{R}^n$ via initialization method.
  - Program which lies closest to network in $\mathbb{R}^n$ is the extracted program.

Could this work?

Conclusions

- 3-layer feedforward networks can approximate $T_p$ for certain programs.
- Using sigmoidal units, the network is trainable using backpropagation.

Open Problems

- How can first-order descriptions be extracted from a connectionist system?
- Can a first-order neural-symbolic system be applied to real world problems, outperforming conventional approaches?
- How does the Core Method relate to reasoning approaches from Cognitive Science?
- ... (many more) ...

www.neural-symbolic.org