Neural-Symbolic Integration
A selfcontained introduction

Sebastian Bader  Pascal Hitzler

ICCL, Technische Universität Dresden, Germany
AIFB, Universität Karlsruhe, Germany
PD Dr. Pascal Hitzler

- Diplom (Mathematics) Univ. of Tübingen 1998
- PhD (Mathematics), Nat. Univ. of Ireland Cork 2001
- 2001-2004 AI Institute TU Dresden
- 2005 Habilitation (Computer Science)
- since 2004 Assistant Professor, AIFB, Univ. of Karlsruhe
  - Knowledge Representation and Reasoning for the **Semantic Web**
  - Neural-Symbolic Integration
  - Mathematical Foundations of Artificial Intelligence
Sebastian Bader

- Diploma (Computer Science) Tech. Univ. Dresden 2003
- PhD Student, TU Dresden 2004 – 2007
- 2007 – 2008 Research Assistant, TU Dresden
- since Aug. 2008 Research Assistant, Dept. Computer Science, University Rostock
New book:

Barbara Hammer, Pascal Hitzler (eds.)
Perspectives of Neural-Symbolic Integration.

With contributions by
Bader, Barreto, de Raedt, Frasconi, Garcez, Geibel, Gust, Hölldobler, Kühnberger, Ritter, Saunders, Seda, Shastri, Sperduti, Tino
Main references for this course


Outline of the Course

▶ Introduction and Motivation
▶ The Core Method for Propositional Logic
▶ Applications and Extensions of the Propositional Core Method
▶ The Core Method for First-Order Logic
▶ More on First-Order & other Perspectives
Part

Introduction and Motivation
Outline: Introduction and Motivation

Motivation

Connectionist Systems

Symbolic AI

Neural-Symbolic Integration

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
Outline: Introduction and Motivation

Motivation

Connectionist Systems

Symbolic AI

Neural-Symbolic Integration
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

- Symbolic System
- Connectionist System
- embedding
- extraction
The Neural Symbolic Cycle

Symbolic System

Connectionist System

embedding

writable

readable

extraction
The Neural Symbolic Cycle

- **Symbolic System**
  - readable
  - writable

- **Connectionist System**
  - extraction
  - embedding
  - trainable

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
Outline: Introduction and Motivation

Motivation

Connectionist Systems
- Introduction
- Applications of Connectionist Systems

Symbolic AI

Neural-Symbolic Integration
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 $\vec{i}$ to the potential $p$:

  $$ p = \sum_n i_n \cdot w_n $$

  $$ p = \sum_n (i_n - w_n)^2 $$

- Output function, mapping the potential $p$ to the output $o$:
Dynamics of a Network

<table>
<thead>
<tr>
<th>Input</th>
<th>Activation F.</th>
<th>Output F.</th>
</tr>
</thead>
<tbody>
<tr>
<td>p set from outside</td>
<td>o = p</td>
<td></td>
</tr>
<tr>
<td>hidden</td>
<td>p = ( \sum_n (i_n - w_n)^2 )</td>
<td>o = e^{-p^2}</td>
</tr>
<tr>
<td>output</td>
<td>p = ( \sum_n (i_n \times w_n) )</td>
<td>o = p</td>
</tr>
</tbody>
</table>
Dynamics of a Network

<table>
<thead>
<tr>
<th></th>
<th>Activation F.</th>
<th>Output F.</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>( p ) set from outside</td>
<td>( o = p )</td>
</tr>
<tr>
<td>hidden</td>
<td>( p = \sum_n (i_n - w_n)^2 )</td>
<td>( o = e^{-p^2} )</td>
</tr>
<tr>
<td>output</td>
<td>( p = \sum_n (i_n * w_n) )</td>
<td>( o = p )</td>
</tr>
</tbody>
</table>

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
Dynamics of a Network

\[
x \rightarrow -1.5 \quad 0.3 \quad 1.8 \quad y \rightarrow -1.3 \quad -0.7 \quad z
\]

\[
t=0: \quad 1.0, -1.0
\]

\[
t=1: \quad 0.74, 0.5783, 0.0001, 2.98
\]

Activation F. | Output F.
---|---
\( p \) set from outside | \( o = p \)
\( p = \sum_n (i_n - w_n)^2 \) | \( o = e^{-p^2} \)
\( p = \sum_n (i_n \times w_n) \) | \( o = p \)
Dynamics of a Network

input: $p$ set from outside
hidden: $p = \sum_n (i_n - w_n)^2$
output: $p = \sum_n (i_n \cdot w_n)$

Activation F.: $o = p$
Output F.: $o = e^{-p^2}$
Dynamics of a Network

Activation function  Output function  Result

\[ p = \sum_n (i_n - w_n)^2 \]
Dynamics of a Network

Activation function  Output function  Result

\[ p = \sum_n (i_n - w_n)^2 \]

\[ p = \sum_n i_n \cdot w_n \]
Training of Connectionist Systems

How can we train a network to represent a function given as a set of samples \((i_1, o_1), \ldots, (i_n, o_n)\)?

Learning as generalization.
Backpropagation

- Let a set of samples \( \{(i_1, o_1), \ldots, (i_n, o_n)\} \) be given.
- Error of the network: \( E = \sum_i (\mathcal{N}(i_i) - o_i)^2 \).
- Idea: minimise \( E \) by gradient descent.
Backpropagation in Detail

1. Present a training sample to the network.
2. Compare the output of the network with the desired output.
3. Calculate the error in each output unit.
4. Modify the weights to the output layer such that the error decreases.
5. Propagate the error back to the last hidden units.
6. Compute the part of the error caused by the hidden units.
7. Modify the weights to the hidden units using this local error.
8. Continue until input units are reached.
A sample run ...

Prior training:

Motivation
Connectionist Systems
Symbolic AI
Neural-Symbolic Integration

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
A sample run ...

Prior training:

After training:
Funahashi’s Theorem

Theorem
Every continuous function $f : K \rightarrow \mathbb{R}$ (with $K \subset \mathbb{R}^n$ compact) can be approximated arbitrarily well using 3 layer feed-forward networks with sigmoidal units (Ken-Ichi Funahashi, 1989).
History of Connectionist Systems

1943  Warren Stirgis 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:

<table>
<thead>
<tr>
<th>Word</th>
<th>phonemes</th>
<th>stress and syllable</th>
</tr>
</thead>
<tbody>
<tr>
<td>logic</td>
<td>ləJɪk</td>
<td>&gt; 1 &lt; 0 &lt;</td>
</tr>
<tr>
<td>programme</td>
<td>proɡr@m--</td>
<td>&gt;&gt; 1 &gt;&gt; 2 &lt;&lt;&lt;&lt;</td>
</tr>
<tr>
<td>neural</td>
<td>nU-r-L</td>
<td>&gt; 1 &lt;&lt; 0 &lt;</td>
</tr>
<tr>
<td>network</td>
<td>nEtw-Rk</td>
<td>&gt; 1 &gt;&gt; 2 &lt;&lt;</td>
</tr>
</tbody>
</table>
ALVNN & MANIAC

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

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)
RALPH - No Hands Across America
TD-Gammon

Artificial Backgammon player *(Gerry Tesauro, 1995)*:

- Based on standard neural network.
- Learns by playing against itself.
- Reaches championship level.
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?
Properties of Connectionist Systems

😊 Massively parallel paradigm.

😊 Continuous function can be approximated arbitrarily well.

😊 Well suited to learn and to adapt to new environments.
  - Hardly any knowledge about the function is needed.
  - Training from raw, incomplete and noisy data.

😊 Degrade gracefully, even if parts of the systems fail the overall behaviour is ok.

😊 Successfully used in many application areas.

😢 Declarative semantics is not available.

😢 Structured objects are hard to represent.

😢 Background knowledge cannot be used.
Outline: Introduction and Motivation

Motivation

Connectionist Systems

Symbolic AI

Introduction
Applications of Symbolic AI

Neural-Symbolic Integration
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 (260).
- Use of logic-like languages for knowledge representation and reasoning.
Motivation

Connectionist Systems

Symbolic AI

Neural-Symbolic Integration

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:

  ```lisp
  ((#IS :B1 #BLOCK))  ((#IS #RED #COLOR))
  ((#IS :B2 #PYRAMID))  ((#IS #GREEN #COLOR))
  ((#IS :B3 #BLOCK))  ((#IS #BLACK #COLOR))
  ((#COLOR :B1 #RED))  ((#CONTAIN :BOX :B4))
  ((#COLOR :B2 #GREEN))  ((#SHAPE :B1 #RECTANGULAR))
  ((#COLOR :TABLE #BLACK))  ((#SHAPE :B3 #RECTANGULAR))
  (DEFPROP TA-AT (THANTE (X Y) (#AT $?X $?Y)
    (THRPLACA (CDR (ATAB $?X)) $?Y))
    THEOREM)
  (DEFPROP TA-EXISTS(THANTE (X) (#EXISTS $?X)
    (THSUCCEED)) THEOREM)
  ```

- 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.
- Phillippè Roussell 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 Jacque 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.
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.
Outline: Introduction and Motivation

Motivation

Connectionist Systems

Symbolic AI

Neural-Symbolic Integration
  Introduction
  The History of Neural-Symbolic Integration
  The Core Method
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 Joint Start - McCulloch and Pitts

Can the activities of a neural system be modelled by a logical calculus?

- A logical calculus of the ideas immanent in nervous activity (W. S. McCulloch and W. Pitts, 1943).
- Representation of events in nerve nets and finite automata (S. C. Kleene, 1956)
Can we model logical connectives using simple units?

Using binary threshold units and the activations 1 for “true” and 0 for “false”, we obtain:

**Disjunction:**

\[ x \lor y \]

**Conjunction:**

\[ x \land y \]

**Negation:**

\[ \neg x \]
McCulloch-Pitts Networks

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**

$I = \{x, y\}$

$U = \{h, o\}$

$O = \{o\}$
Moore Machines

- A Moore Machine consists of:
  - set of states with an initial state
  - set of input symbols
  - set of output symbols
  - state transition function
  - state output function
Moore Machines

- A Moore Machine consists of:
  - set of states with an initial state
  - set of input symbols
  - set of output symbols
  - state transition function
  - state output function

- Using the Moore-machine:
  Input: \( a \ b \ b \ a \)
  State: \( q_0 \qquad q_1 \)
  Output:
Moore Machines

- A Moore Machine consists of:
  - set of states with an initial state
  - set of input symbols
  - set of output symbols
  - state transition function
  - state output function

- Using the Moore-machine:
  Input: a b b a
  State: q₀ → q₁ → q₀ → q₁
  Output: 1
**Moore Machines**

- A Moore Machine consists of:
  - set of states with an initial state
  - set of input symbols
  - set of output symbols
  - state transition function
  - state output function

- Using the Moore-machine:
  Input: $a$ $b$ $b$ $a$
  State: $q_0$ $q_1$ $q_0$ $q_1$ $q_0$ $q_1$
  Output: $1$ $0$
Moore Machines

- A Moore Machine consists of:
  - set of states with an initial state
  - set of input symbols
  - set of output symbols
  - state transition function
  - state output function

- Using the Moore-machine:
  Input:
  
  State:
  
  Output:
Moore Machines

- A Moore Machine consists of:
  - set of states with an initial state
  - set of input symbols
  - set of output symbols
  - state transition function
  - state output function

- Using the Moore-machine:

  **Input:**
  \[ a \ b \ b \ a \]

  **State:**
  \[
  \begin{array}{c}
  q_0 \rightarrow q_1 \\
  a \rightarrow b \\
  b \rightarrow b \\
  a \rightarrow a \\
  \end{array}
  \]

  **Output:**
  \[ 1 \ 0 \ 1 \ 1 \]
From Moore Machines to McCulloch-Pitts Networks
From Moore Machines to McCulloch-Pitts Networks
From Moore Machines to McCulloch-Pitts Networks
From Moore Machines to McCulloch-Pitts Networks
From Moore Machines to McCulloch-Pitts Networks
From Moore Machines to McCulloch-Pitts Networks

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

▶ A sample network

▶ Moore Machine:
  - Set of states \( (Q) \)
  - Input symbols \( (\Sigma) \)
  - Output symbols \( (\Delta) \)
  - State transitions \( (\delta) \)
  - State outputs \( (\lambda) \)
From McCulloch-Pitts Networks to Moore Machines

- **A sample network**

- **Moore Machine:**
  - Set of states \((Q)\)
  - Input symbols \((\Sigma)\)
  - Output symbols \((\Delta)\)
  - State transitions \((\delta)\)
  - State outputs \((\lambda)\)

\[ Q = \{ [0, 0, 0, 0], [0, 0, 0, 1], [0, 0, 1, 0], [0, 0, 1, 1], [0, 1, 0, 0], [0, 1, 0, 1], [0, 1, 1, 0], [0, 1, 1, 1] \} \]
From McCulloch-Pitts Networks to Moore Machines

- A sample network
  ![A sample network](image)

- Moore Machine:
  - Set of states \( Q \)
  - Input symbols \( \Sigma \)
  - Output symbols \( \Delta \)
  - State transitions \( \delta \)
  - State outputs \( \lambda \)

\[
Q = \{ \text{states} \} \\
\Sigma = \{ \text{input symbols} \}
\]
From McCulloch-Pitts Networks to Moore Machines

- A sample network

- Moore Machine:
  - Set of states \((Q)\)
  - Input symbols \((\Sigma)\)
  - Output symbols \((\Delta)\)
  - State transitions \((\delta)\)
  - State outputs \((\lambda)\)
From McCulloch-Pitts Networks to Moore Machines

A sample network

Moore Machine:
- Set of states ($Q$)
- Input symbols ($\Sigma$)
- Output symbols ($\Delta$)
- State transitions ($\delta$)
- State outputs ($\lambda$)

$Q = \{ \begin{array}{c} \circ \circ \\
\bullet \circ \\
\circ \bullet \\
\bullet \bullet \end{array} \}$

$\Sigma = \{ \begin{array}{c} \circ \circ \\
\bullet \circ \\
\circ \bullet \\
\bullet \bullet \end{array} \}$

$\Delta = \{ \begin{array}{c} \circ \circ \\
\bullet \bullet \end{array} \}$

$\delta : Q \times \Sigma \rightarrow Q$

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
From McCulloch-Pitts Networks to Moore Machines

▶ A sample network

Moore Machine:
- Set of states ($Q$)
- Input symbols ($\Sigma$)
- Output symbols ($\Delta$)
- State transitions ($\delta$)
- State outputs ($\lambda$)

$Q = \{\text{state1, state2, state3, state4}\}$
$\Sigma = \{\text{symbol1, symbol2, symbol3}\}$
$\Delta = \{\text{output1, output2}\}$
$\delta : Q \times \Sigma \rightarrow Q$
$\lambda : Q \rightarrow \Delta$

$\lambda(Q) = \{\text{output1, output2, output3, output4}\}$
Conclusions

- McCulloch-Pitts networks are finite automata and vice versa.
- Similar constructions work for other types of automata.
- The paper ("A logical calculus of the ideas immanent in nervous activity") started the research on artificial neural networks and on finite automata.
Humans can handle certain problems very easily and fast.
Humans approximate knowledge base: $10^8$ rules and facts, i.e. we perform reflexive reasoning in sublinear time.
The SHRUTI-System (Shastri & Ajjanagadde, 1993) is a connectionist architecture for this type of reasoning.
Variable binding by synchronization of neurons.
SHRUTI - A Sample Knowledge Base

► Rules:

\[
\text{owns}(Y, Z) \leftarrow \text{gives}(X, Y, Z)\\
\text{owns}(X, Y) \leftarrow \text{buys}(X, Y)\\
\text{can} \neg \text{sell}(x, y) \leftarrow \text{owns}(X, Y)
\]

► Facts:

\[
\text{gives}(\text{john}, \text{josephine}, \text{book})\\
(\exists X)\text{buys}(\text{john}, X)\\
\text{owns}(\text{josephine}, \text{ball})
\]

► Question:

\[
\text{can} \neg \text{sell}(\text{josephine}, \text{book})? \quad \text{yes}\\
(\exists X)\text{owns}(\text{josephine}, X)? \quad \text{yes} \ (X \mapsto \text{book}, X \mapsto \text{ball})
\]
SHRUTI - A Sample Network

Can-sell

Owns

Gives

Buys

from john

from josephine

from book

from john

from john

from book

from john
SHRUTI - A Sample Network Run

buys ▲
buys 2nd arg
buys 1st arg
buys ▼
buys △
gives ▲
gives 3nd arg
gives 2nd arg
gives 1st arg
gives ▼
gives △
owns ▲
owns 2nd arg
owns 1st arg
owns ▼
owns △
can-sell 2nd arg
can-sell 1st arg
can-sell ▼
can-sell △

josephine
ball
john
book
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.

Support of negation and inconsistency (Shastri & Wendelken, 1999).


Multiple instantiation of a single rule (Wendelken & Shastri, 2004).
The Core Method

- Relate logic programs and connectionist systems
The Core Method

- Relate logic programs and connectionist systems
- Embed interpretations into (vectors of) real numbers.
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.
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 $f_P$. 

\[ \begin{align*}
I_L & \xrightarrow{T_P} I_L \\
\downarrow \iota & \quad & \uparrow \iota^{-1} \\
\mathbb{R}^m & \xrightarrow{f_P} \mathbb{R}^m
\end{align*} \]
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 $\iota$?)
- How can symbolic knowledge be *extracted* from connectionist systems? (What is $\iota^{-1}$?)
- How can symbolic knowledge be *learned* using connectionist systems?
- How can connectionist *learning be guided* by symbolic background knowledge?
Outline of the Course

- Introduction and Motivation
- The Core Method for Propositional Logic
- Applications and Extensions of the Propositional Core Method
- The Core Method for First-Order Logic
- More on First-Order & other Perspectives