Neural-Symbolic Integration
A selfcontained introduction

Sebastian Bader  Pascal Hitzler

ICCL, Technische Universität Dresden, Germany
AIFB, Universität Karlsruhe, Germany
Outline of the Course

- Introduction and Motivation
- The Core Method for Propositional Logic
- Applications of the Propositional Core Method
- A New Approach to Pedagogical Extraction
- The Core Method for First-Order Logic
- More on First-Order & other Perspectives
Part

The Core Method for First Order Logic
Outline: The Core Method for First Order Logic

Introduction

Feasibility

The FineBlend System

Further Topics

Conclusions
Outline: The Core Method for First Order Logic

Introduction
The Core Method
First Order Logic Programs

Feasibility

The FineBlend System

Further Topics

Conclusions
The Core Method

- Relate logic programs and connectionist systems

\[ I_L \xrightarrow{T_P} I_L \]
The Core Method

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

\[
\begin{align*}
I_L & \xrightarrow{T_P} I_L \\
\downarrow \iota & \quad \uparrow \iota^{-1} \\
\mathbb{R}^m & \quad \mathbb{R}^m
\end{align*}
\]
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.
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 $f_P$. 

![Diagram showing the core method](image_url)
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.
First Order Logic Programs – Two Examples

\[\text{nat}(0).\]
\[\text{nat}(\text{succ}(X)) \leftarrow \text{nat}(X).\]
\[\text{even}(0).\]
\[\text{even}(\text{succ}(X)) \leftarrow \text{odd}(X).\]
\[\text{odd}(X) \leftarrow \neg\text{even}(X).\]

% 0 is a natural number.  
% The successor \text{succ}(X) is a natural number if \text{X} is a natural number.  
% 0 is an even number.  
% The successor of an odd \text{X} is even.  
% If \text{X} is not even then it is odd.
First Order Logic Programs – The Syntax

Functions, Variables and Terms
\[ F = \{0/0, \text{succ}/1\} \]
\[ V = \{X\} \]
\[ T = \{0, \text{succ}(0), \text{succ}(X), \text{succ}(	ext{succ}(0)), \ldots\} \]

Predicate Symbols and Atoms
\[ P = \{\text{even}/1, \text{odd}/1\} \]
\[ A = \{\text{even}(	ext{succ}(X)), \text{odd}(	ext{succ}(0)), \text{odd}(0), \text{odd}(X), \ldots\} \]

Connectives, Clause and Program
\[ \text{propositional logic} \]
First Order Logic Programs – The Semantics

**Herbrand Base** $B_L = \text{Set of ground atoms}$

\[ B_L = \{ \text{even}(0), \text{even}(\text{succ}(0)), \ldots, \text{odd}(0), \text{odd}(\text{succ}(0)), \ldots \} \]

**Interpretations = Subsets of the Herbrand base**

\[ I_1 = \{ \text{even}(\text{succ}^{2n}(0) \mid n \geq 1 \} \quad I_2 = \{\} \]

\[ I_3 = \{ \text{odd}(\text{succ}^{2n+1}(0) \mid n \geq 0 \} \quad I_4 = I_1 \cup I_3 \]

\[ I_L = \text{Space of all Interpretations} \]
$T_P$ for our running examples

**Definition ($T_P$)**

$$T_P(I) = \{ A \mid \text{there is } A \leftarrow \text{body in } \text{ground}(P) \text{ and } I \models \text{body} \}$$

**Example (Natural numbers)**

$\{\}\mapsto\{n(0)\}$

$n(0)$.

$\{n(0)\}\mapsto\{n(0), n(s(0))\}$

$n(s(X)) \leftarrow n(X)$.

$\{n(0), n(s(0))\}\mapsto\{n(0), n(s(0)), n(s(s(0)))\}$

$\{n(X) \mid X \in T\}\mapsto\{n(X) \mid X \in T\}$

**Example (Even and odd numbers)**

$\{\}\mapsto\{e(0), o(X) \mid X \in T\}$

$e(0)$.

$\{o(X) \mid X \in T\}\mapsto\{e(0), e(s(X))\}$

$e(s(X)) \leftarrow o(X)$.

$\{e(s(2n(0))) \mid n \geq 0\}\mapsto\{e(0), o(s(2n(0)+1)) \mid n \geq 0\}$

$o(X) \leftarrow \neg e(X)$.

$\{o(s(2n(0)+1)) \mid n \geq 0\}\mapsto\{e(0), e(s(2n(0))) \mid n \geq 0\}$

$BL \mapsto\{e(0), e(s(X)) \mid X \in T\}$
**Definition** ($T_P$)

$$T_P(I) = \{ A \mid \text{there is } A \leftarrow \text{body in } \text{ground}(P) \text{ and } I \models \text{body} \}$$

**Example (Natural numbers)**

\[
\begin{align*}
&\{\} \mapsto \{n(0)\} \\
&\{n(0)\} \mapsto \{n(0), n(s(0))\} \\
&\{n(0), n(s(0))\} \mapsto \{n(0), n(s(0)), n(s(s(0)))\} \\
&\{n(X) \mid X \in T\} \mapsto \{n(X) \mid X \in T\}
\end{align*}
\]

**Example (Even and odd numbers)**

\[
\begin{align*}
&\{\} \mapsto \{e(0), o(X) \mid X \in T\} \\
&o(X) \mid X \in T \mapsto \{e(0), e(s(X)), o(X) \mid X \in T\} \\
&e(s(X)) \leftarrow o(X). \\
&\{e(s^{2n}(0)) \mid n \geq 0\} \mapsto \{e(0), o(s^{2n+1}(0)) \mid n \geq 0\} \\
&o(X) \leftarrow \neg e(X). \\
&\{o(s^{2n+1}(0)) \mid n \geq 0\} \mapsto \{e(0), e(s^{2n}(0)) \mid n \geq 0\} \\
&B_L \mapsto \{e(0), e(s(X)) \mid X \in T\}
\end{align*}
\]
Problems

- $B_\mathcal{L}$ is usually infinite and therefore the propositional approach does not work.
- Is the Core method applicable?
- How can we bridge the gap?
Outline: The Core Method for First Order Logic

Introduction

Feasibility

The FineBlend System

Further Topics

Conclusions

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
Level Mappings

- A *Level Mapping* assigns a (unique) natural number to each ground atom ...

*Example (Even and odd numbers)*

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

- A Level Mapping assigns a (unique) natural number to each ground atom ...

Example (Even and odd numbers)

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

- ... hence, enumerates the Herbrand base:

Example (Even and odd numbers)

\[
[e(0), o(0), e(s(0)), o(s(0)), e(s(s(0))), \ldots]
\]

\[
\begin{align*}
1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 \\
\end{align*}
\]
Embedding First-Order Terms into the Real Numbers

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

$$\iota(l) = \sum_{A \in l} 4^{-|A|}$$

This coincides with a “binary” representation:

$$B_{\mathcal{L}} = [ e(0), o(0), e(1), o(1), e(2), \ldots ]$$
Embedding First-Order Terms into the Real Numbers

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

\[ \iota(I) = \sum_{A \in I} 4^{-|A|} \]

This coincides with a “binary” representation:

\[ B_L = [e(0), o(0), e(1), o(1), e(2), \ldots] \]

\[ \iota(\{e(0)\}) = 0.100004 \]
Embedding First-Order Terms into the Real Numbers

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

\[ \nu(I) = \sum_{A \in I} 4^{-|A|} \]

This coincides with a “binary” representation:

\[ B_L = [ e(0), o(0), e(1), o(1), e(2), \ldots ] \]

\[ \nu(\{ e(0) \}) = 0.100004 \approx 0.25_{10} \]
Embedding First-Order Terms into the Real Numbers

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

\[ \nu(I) = \sum_{A \in I} 4^{-|A|} \]

This coincides with a “binary” representation:

\[ B_L = [ e(0), o(0), e(1), o(1), e(2), \ldots ] \]

\[ \nu(\{e(0)\}) = 0.10000_4 = 0.25_{10} \]

\[ \nu(\{e(0), e(1), e(2)\}) = 0.10101_4 \approx 0.27_{10} \]
$\mathcal{C}$ - The Set of all embedded Interpretations

- $\mathcal{C}$ for the 1-dimensional case:

$$\mathcal{C} = \{\nu(l) \mid l \in l_\mathcal{L}\}$$
\mathcal{C} - The Set of all embedded Interpretations

▶ \mathcal{C} for the 1-dimensional case:

\[ \mathcal{C} = \{ \iota(I) \mid I \in I_L \} \]

▶ Another construction:
The Graph of the Natural Numbers

\[ n(0). \]
\[ n(s(X)) \leftarrow n(X). \]
\[ |n(s^n(0))| = n + 1 \]
The Graph of the Natural Numbers

\[ \nu(T_P(1)) \]

\[
n(0).
n(s(X)) \leftarrow n(X).
\]

\[
|n(s^n(0))| = n + 1
\]
The Graph of the Natural Numbers

\[ \nu(T_P(I)) \]

\[ n(0). \]
\[ n(s(X)) \leftarrow n(X). \]

\[ |n(s^n(0))| = n + 1 \]

\[ \{ \} \mapsto \{ n(0) \} \]

\[ \{ n(0) \} \mapsto \{ n(0), n(s(0)) \} \]

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
The Graph of the Even and Odd Numbers

\[ \nu(T_P(I)) \]

\[ e(0). \]
\[ e(s(X)) \leftarrow o(X). \]
\[ o(X) \leftarrow \neg e(X). \]
Continuity of the Embedded $T_P$-Operator

- The mapping $\iota$ is ...
  - is continuous
  - is a bijection between $I_\mathcal{L}$ and $\mathcal{C}$
  - has a continuous inverse $\iota^{-1}$
  - hence, is a homeomorphism

- The set $I_\mathcal{L}$ together with a suitable metric is compact $\leadsto \mathcal{C}$ is compact.

- $T_P$ is continuous for certain programs $\leadsto f_P$ is continuous

- We can apply Funahashis theorem.
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.
Approximating the Embedded $T_P$-Operator

Constructions using sigmoidal and RBF-units are given in (Bader, Hitzler & Witzel, 2005).
Approximating the Embedded $T_P$-Operator

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

\[ e(0). \]
\[ e(s(X)) \leftarrow o(X). \]
\[ o(X) \leftarrow \neg e(X). \]
\[ \varepsilon = 0.05 \]
Approximating the Embedded $T_P$-Operator

Constructions using sigmoidal and RBF-units are given in (Bader, Hitzler & Witzel, 2005).
Approximating the Embedded $T_P$-Operator

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

$$\iota(T_P(I))$$

$$e(0).$$
$$e(s(X)) \leftarrow o(X).$$
$$o(X) \leftarrow \neg e(X).$$

$$\varepsilon = 0.05$$
Approximating the Embedded $T_P$-Operator

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.
Outline: The Core Method for First Order Logic

Introduction

Feasibility

The FineBlend System
  Multi-Dimensional Approach
  The FineBlend System

Further Topics

Conclusions
Multi-dimensional Level Mappings

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

**Example (Even and odd numbers)**

$$\| e(s^n(0)) \| = (n + 1, 1) \quad \| o(s^n(0)) \| = (n + 1, 2)$$
Multi-dimensional Level Mappings

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

**Example (Even and odd numbers)**

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

► ... still “enumerates” the Herbrand base:

**Example (Even and odd numbers)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dim1</em></td>
<td><em>e</em>(0)</td>
<td><em>e</em>(s(0))</td>
<td><em>e</em>(s(s(0)))</td>
<td><em>e</em>(s(s(s(0))))</td>
</tr>
<tr>
<td><em>dim2</em></td>
<td><em>o</em>(0)</td>
<td><em>o</em>(s(0))</td>
<td><em>o</em>(s(s(0)))</td>
<td><em>o</em>(s(s(s(0))))</td>
</tr>
</tbody>
</table>
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:

$$\vec{ι}(l) = \sum_{A \in l} \vec{ι}(A)$$

$$\vec{ι}(A) = (ι_1(A), \ldots, ι_m(A))$$ with

$$ι_i(A) = \begin{cases} 4^{-l} & \text{for } \|A\| = (l, d) \text{ and } i = d \\ 0 & \text{otherwise} \end{cases}$$
\( \mathcal{C} \) - The Set of all embedded Interpretations

- \( \mathcal{C} \) for the 2-dimensional case:

\[
\mathcal{C} = \{ \vec{\iota}(I) \mid I \in \mathcal{I} \}
\]

- Another construction:
\( \mathcal{C} - \text{The Set of all embedded Interpretations} \)

- \( \mathcal{C} \) for the 2-dimensional case:

\[
\mathcal{C} = \{ \overline{\mathcal{I}}(l) \mid l \in I_\mathcal{C} \}
\]

- Another construction:
\( \mathcal{C} \) - The Set of all embedded Interpretations

- **\( \mathcal{C} \)** for the 2-dimensional case:

\[
\mathcal{C} = \{ \vec{\iota}(l) \mid l \in \mathcal{I}_C \}
\]

- Another construction:

\[
\begin{array}{cc}
\{e(0)\} & \{e(0), o(0)\} \\
\{} & \{e(0)\}
\end{array}
\]
\[ \mathcal{C} - \text{The Set of all embedded Interpretations} \]

- For the 2-dimensional case:

\[ \mathcal{C} = \{ \vec{\iota}(I) \mid I \in I_L \} \]

- Another construction:
\( \mathcal{C} \) - The Set of all embedded Interpretations

\( \mathcal{C} \) for the 2-dimensional case:

\[
\mathcal{C} = \{ \bar{\iota}(l) \mid l \in \mathcal{I} \}
\]

Another construction:

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
- The Set of all embedded Interpretations

- for the 2-dimensional case:

\[
\mathcal{C} = \{ \vec{\iota}(l) \mid l \in \mathcal{I_c} \}
\]

- Another construction:
Approximating the Embedded $T_P$-Operator

\[ \iota(T_P(I)) \]

\[ \iota(I) \]

\[ d_2 \]

\[ d_1 \]
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 FineBlend

- Incoming weights denote areas of responsibility
- Winner-take-all hidden layer
- Outgoing weights denote value of the function

```
0.3
0.3
```

```
\text{"input layer connections"}
\text{"output layer connections"}
```
FineBlend: Training

- Incoming weights encode *responsible areas*:
FineBlend: Training

- Incoming weights encode *responsible areas*:

- Adapting the input weights:
FineBlend: Adding New Units

- Units accumulate the error caused in their area.
- If this exceeds a certain threshold ...:
  - The unit is moved to the centre of its responsible area.
  - A new unit is inserted into one sub-area.
FineBlend: Removing Inutile Units

- Every unit maintains a utility value.
- If this drops below a given threshold, the corresponding unit is removed:
Statistics - FineBlend vs SGNG

The chart compares the performance of FineBlend and SGNG in terms of error and number of units. The x-axis represents the number of examples, while the y-axes show the error and the number of units.

Key points:
- FineBlend (solid line) generally has lower error compared to SGNG (dotted line).
- The number of units required for FineBlend is consistently lower than that for SGNG.

This comparison highlights the efficiency and accuracy of FineBlend in neural-symbolic integration tasks.
Statistics - Unit Failure
Statistics - Iteration of Random Inputs
Conclusions

😊 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.
Outline: The Core Method for First Order Logic

Introduction

Feasibility

The FineBlend System

Further Topics

Conclusions
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
  \[ d(I, I_n) < \varepsilon. \]
- Use propositional approach to encode $P_n$.
- Increasing $n$ yields better approximations of $T_P$. (If $T_P$ is continuous wrt. $d$.)
- 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:
Iterated Function Symbols

The Sierpinsky Triangle:

```
Iterated Function Symbols

The Sierpinsky Triangle:

The Sierpinsky Triangle:
```

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
Iterated Function Symbols

The Sierpinski Triangle:
Iterated Function Symbols

The Sierpinsky Triangle:

$$\begin{align*}
\text{The Sierpinsky Triangle:} & \\
\end{align*}$$
Iterated Function Symbols

▶ The Sierpinsky Triangle:
Iterated Function Symbols

▶ The Sierpinsky Triangle:
From Logic Programs to Iterated Function Systems

- For some logic programs we can explicitly construct an IFS, such that the attractor coincides with the graph of the embedded $T_P$-operator.
- Let $P$ be a program such that $f_P$ is Lipschitz-continuous. Then there exists an IFS such that the attractor is the graph of $f_P$.
- 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.
Outline: The Core Method for First Order Logic

Introduction

Feasibility

The FineBlend System

Further Topics

Conclusions
Conclusions

- 3-layer feedforward networks can approximate $T_P$ for certain programs. i.e., the Core method is applicable.
- Using sigmoidal units, the network is trainable using backpropagation.
- One-dimensional approach requires high accuracy.
- Using vector-based networks we can achieve higher accuracy.
- Are there applications for the first-order case?
- Can we outperform other systems?
Outline of the Course

- Introduction and Motivation
- The Core Method for Propositional Logic
- Applications of the Propositional Core Method
- A New Approach to Pedagogical Extraction
- The Core Method for First-Order Logic
- More on First-Order & other Perspectives