Combining Logic and Connectionist Systems

Steffen Hölldobler
International Center for Computational Logic
Technische Universität Dresden
Germany

- Introduction & Motivation
- The Very Idea
- Propositional Logic Programs
- First-Order Logic Programs
- Challenge Problems

"Logic is everywhere ..."
Introduction & Motivation: Connectionist Systems

- Well-suited to learn, to adapt to new environments, to degrade gracefully etc.
- Many successful applications.
- Approximate functions.
  - Hardly any knowledge about the functions is needed.
  - Trained using incomplete data.
- Declarative semantics is not available.
- Recursive networks are hardly understood.
- We still observe a propositional fixation [McCarthy:88].
- Structured objects are difficult to represent.

Can we instantiate the power of symbolic computation within fully connectionist systems? [Smolensky:87]
Introduction & Motivation: Logic Systems

- Well-suited to represent and reason about structured objects and structure-sensitive processes.
- Many successful applications.
- Direct implementation of relations and functions.
- Explicit expert knowledge is required.
- Highly recursive structures.
- Well understood declarative semantics.
- Logic systems are brittle.
- Expert knowledge may not be available.

Can we instantiate the power of connectionist computation within a logic system?
Introduction & Motivation: Objective

► Seek the best of both paradigms!
► Understanding the relation between connectionist and logic systems.
► Contribute to the open research problems of both areas.
► Well developed for propositional case.
► Hard problem: going beyond.
► In this talk: logic programs.

Semantic operators for logic programs can be computed by connectionist systems.
The Very Idea

- Various semantics for logic programs coincide with fixed points of associated immediate consequence operators [Apt&vanEmden:82].

- **Banach Contraction Mapping Theorem** A contraction mapping $f$ defined on a complete metric space $(X, d)$ has a unique fixed point. The sequence $y, f(y), f(f(y)), \ldots$ converges to this fixed point for any $y \in X$.

  Consider logic programs, whose immediate consequence operator is a contraction [Fitting:94].

- Every continuous function on the reals can be uniformly approximated by feedforward connectionist networks [Funahashi:89].

  Consider logic programs, whose immediate consequence operator is continuous on the reals [H&Kalinke&Störr:99].
Logic Programs

- A logic program $P$ over a first-order language $\mathcal{L}$ is a finite set of clauses
  \[ A \leftarrow L_1 \land \ldots \land L_n, \]
  where $A$ is an atom, $L_i$ are literals and $n \geq 0$.
- $B_{\mathcal{L}}$ is the set of all ground atoms over $\mathcal{L}$ called Herbrand base.
- A Herbrand interpretation $I$ is a mapping $B_{\mathcal{L}} \rightarrow \{\top, \bot\}$.
- $2^{B_{\mathcal{L}}}$ is the set of all Herbrand interpretations.
- $\text{ground}(P)$ is the set of all ground instances of clauses in $P$.
- Immediate consequence operator $T_P : 2^{B_{\mathcal{L}}} \rightarrow 2^{B_{\mathcal{L}}}$:
  \[ T_P(I) = \{A \mid \text{there is a clause } A \leftarrow L_1 \land \ldots \land L_n \in \text{ground}(P) \]
  \[ \text{such that } I \models L_1 \land \ldots \land L_n\} \].
- $I$ is a supported model iff $T_P(I) = I$. 

Combining Logic and Connectionist Systems
3-Layer Recurrent Networks

output layer

hidden layer

input layer
3-Layer Recurrent Networks

- Input layer
- Hidden layer
- Output layer

(kernel)
3-Layer Recurrent Networks
3-Layer Recurrent Networks

![Diagram of a 3-layer recurrent network with input layer, hidden layer, and output layer connected by kernel connections.]

- At each point in time all units do:
  - compute sum of their weighted inputs,
  - apply linear, threshold or sigmoidal function to obtain output.
Propositional Logic Programs

- Given a logic program $P$, e.g. $P = \{ A, C \leftarrow A \land \neg B, C \leftarrow \neg A \land B \}$.
- A translation algorithm translates $P$ into a kernel of binary threshold units:

\[
\begin{array}{ccc}
A & B & C \\
0.5 & 0.5 & 0.5 \\
\end{array}
\]

- output layer

- hidden layer

\[
\begin{array}{ccc}
A & B & C \\
0.5 & 0.5 & 0.5 \\
\end{array}
\]

- input layer
Propositional Logic Programs

- Given a logic programs \( P \), e.g. \( P = \{ A, C ← A ∧ ¬B, C ← ¬A ∧ B \} \).
- A translation algorithm translates \( P \) into a kernel of binary threshold units:

\[
\begin{array}{ccc}
A & B & C \\
0.5 & 0.5 & 0.5 \\
\end{array}
\]

\[
\begin{array}{c}
\text{output layer} \\
1 \\
\end{array}
\]

\[
\begin{array}{c}
\text{hidden layer} \\
-0.5 \\
\end{array}
\]

\[
\begin{array}{ccc}
A & B & C \\
0.5 & 0.5 & 0.5 \\
\end{array}
\]

\[
\begin{array}{c}
\text{input layer} \\
\end{array}
\]
Propositional Logic Programs

- Given a logic program $P$, e.g. $P = \{A, C \leftarrow A \land \neg B, C \leftarrow \neg A \land B\}$.
- A translation algorithm translates $P$ into a kernel of binary threshold units:

```
  +-------+-------+-------+  
  | 0.5   | 0.5   | 0.5   |  
  +-------+-------+-------+  
  | 1      |       |       |  
  +-------+-------+-------+  
  | -0.5   | 0.5   |       |  
  |       |       |       |  
  +-------+-------+-------+  
  |       |       |       |  
  +-------+-------+-------+  
  | 0.5   | 0.5   | 0.5   |  
  +-------+-------+-------+  
```

output layer

hidden layer

input layer
Propositional Logic Programs

▶ Given a logic program $P$, e.g. $P = \{A, C \leftarrow A \land \neg B, C \leftarrow \neg A \land B\}$.
▶ A translation algorithm translates $P$ into a kernel of binary threshold units:
Some Results for the Propositional Case

- **Proposition** 2-layer networks cannot compute $T_P$ for definite $P$.
- **Theorem** For each program $P$, there exists a kernel computing $T_P$.
- A logic program $P$ is said to be **strongly determined** if there exists a metric $d$ on the set of all Herbrand interpretations for $P$ such that $T_P$ is a contraction wrt $d$.
- **Corollary** Let $P$ be a strongly determined program. Then there exists a 3-layer recurrent network such that the computation with an arbitrary initial input converges and yields the unique fixed point of $T_P$.
- Let $n$ be the number of clauses and $m$ be the number of propositional variables occurring in $P$.
  - $2m + n$ units, $2mn$ connections in the kernel.
  - $T_P(I)$ is computed in 2 steps.
  - The parallel computational model to compute $T_P(I)$ is optimal.
  - The recurrent network settles down in $3n$ steps in the worst case.
Extensions

- Replace threshold units by sigmoidal ones [Garcez&Zaverucha&Carvalho:97].
  - Relation to logic programs is preserved.
  - Kernels can now be trained using e.g. backpropagation.
Extensions

- Replace threshold units by sigmoidal ones [Garcez&Zaverucha&Carvalho:97].
  - Relation to logic programs is preserved.
  - Kernels can now be trained using e.g. backpropagation.

- Apply rule extraction techniques to the trained kernels.
  - Refined logic programs; Knowledge based artificial neural networks.
Extensions

► Replace threshold units by sigmoidal ones [Garcez&Zaverucha&Carvalho:97].
  ▶ Relation to logic programs is preserved.
  ▶ Kernels can now be trained using e.g. backpropagation.

► Apply rule extraction techniques to the trained kernels.
  ▶ Refined logic programs; Knowledge based artificial neural networks.

► Propositional modal programs [Garcez&Lamb&Gabbay:2002].
  ▶ Introduce modalities $\square$ and $\Diamond$ as well as relations between worlds.
  ▶ Refine translation algorithm such that the immediate consequence operator is again computed by a kernel.
  ▶ For each world, turn the kernel into a recurrent network.
  ▶ Connect worlds with respect to the given set of relations and the Kripke semantics of the modalities.
First Order Logic Programs

- Given a logic program $P$ over a first order language $\mathcal{L}$ together with $T_P : 2^{B_\mathcal{L}} \rightarrow 2^{B_\mathcal{L}}$.
- $B_\mathcal{L}$ is countably infinite.
- The method used to relate propositional logic and connectionist systems is not applicable.

How can the gap between the discrete, symbolic setting of logic, and the continuous, real valued setting of connectionist networks be closed?
The Goal

- Find $\iota : 2^B \to \mathbb{R}$ and $f_P : \mathbb{R} \to \mathbb{R}$ such that the following conditions hold.
  - $T_P(I) = J$ implies $f_P(\iota(I)) = \iota(J)$.
  - $f_P(r) = s$ implies $T_P(\iota^{-1}(r)) = \iota^{-1}(s)$.

  $\Rightarrow$ $f_P$ is a sound and complete encoding of $T_P$.

- $T_P$ is a contraction on $2^B \iff f_P$ is a contraction on $\mathbb{R}$.

  $\Rightarrow$ The contraction property and fixed points are preserved.

- $f_P$ is continuous on $\mathbb{R}$.

  $\Rightarrow$ A connectionist network approximating $f_P$ is known to exist.
Acyclic Logic Programs

- Let $P$ be a program over a first order language $\mathcal{L}$.
- A level mapping for $P$ is a function $l : B_\mathcal{L} \to \mathbb{N}$.
  - We define $l(\neg A) = l(A)$.
- We can associate a metric $d_\mathcal{L}$ with $\mathcal{L}$ and $l$. Let $I, J \in 2^{B_\mathcal{L}}$:
  \[
d_\mathcal{L}(I, J) = \begin{cases} 
0 & \text{if } I = J \\
2^{-n} & \text{if } n \text{ is the smallest level on which } I \text{ and } J \text{ differ.}
\end{cases}
\]
- Proposition $(2^{B_\mathcal{L}}, d_\mathcal{L})$ is a complete metric space [Fitting:94].
- $P$ is said to be acyclic wrt a level mapping $l$, if for every $A \leftarrow L_1 \land \ldots \land L_n \in \text{ground}(P)$ we find $l(A) > l(L_i)$ for all $i$.
- Proposition Let $P$ be an acyclic logic program wrt $l$ and $d_\mathcal{L}$ the metric associated with $\mathcal{L}$ and $l$, then $T_P$ is a contraction on $(2^{B_\mathcal{L}}, d_\mathcal{L})$. 
Mapping Interpretations to Real Numbers

- Let $\mathcal{D} = \{ r \in \mathbb{R} \mid r = \sum_{i=1}^{\infty} a_i 4^{-i}, \text{ where } a_i \in \{0, 1\} \text{ for all } i \}$. 
- Let $\ell$ be a bijective level mapping. 
- $\{\top, \bot\}$ can be identified with $\{0, 1\}$. 
- The set of all mappings $I: B_L \to \{\top, \bot\}$ can be identified with the set of all mappings $f: \mathbb{N} \to \{0, 1\}$. 
- Let $I_L$ be the set of all mappings from $B_L$ to $\{0, 1\}$. 
- Let $\iota: I_L \to \mathcal{D}$ be defined as
  \[ \iota(I) = \sum_{i=1}^{\infty} I(\ell^{-1}(i)) 4^{-i}. \]
- Proposition $\iota$ is a bijection. 

We have a sound and complete encoding of interpretations.
We define \( f_P : D \rightarrow D : r \mapsto \nu(T_P(\nu^{-1}(r))) \). We have a sound and complete encoding of \( T_P \).

**Proposition** Let \( P \) be an acyclic program wrt a bijective level mapping. \( f_P \) is a contraction on \( D \). Contraction property and fixed points are preserved.
Approximating Continuous Functions

- **Corollary** $f_P$ is continuous.

- **Theorem** [Funahashi:89] Suppose that $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is non-constant, bounded, monotone increasing and continuous. Let $K \subseteq \mathbb{R}^n$ be compact, let $f : K \rightarrow \mathbb{R}$ be continuous, and let $\varepsilon > 0$. Then there exists a 3-layer feed forward network with sigmoidal function $\phi$ for the hidden layer and linear activation function for the input and output layer whose input-output mapping $\overline{f} : K \rightarrow \mathbb{R}$ satisfies

  \[
  \max_{x \in K} |f(x) - \overline{f}(x)| < \varepsilon.
  \]

  Every continuous function $f : K \rightarrow \mathbb{R}$ can be uniformly approximated by input-output functions of 3-layer feed forward networks.

- **Theorem** $f_P$ can be uniformly approximated by input-output functions of 3-layer feed forward networks.

  $T_P$ can be approximated as well by applying $\iota^{-1}$.

Connectionist network approximating immediate consequence operator exists.
An Example

Consider \( P = \{ q(0), \, q(s(X)) \leftarrow q(X) \} \) and let \( l(q(s^n(0))) = n + 1 \).

- \( P \) is acyclic wrt \( l \), \( l \) is bijective, \( \iota(B_L) = \frac{1}{3} \).
- \( f_P(\iota(I)) = 4^{-l(q(0))} + \sum_{q(x) \in I} 4^{-l(q(s(X)))} = 4^{-l(q(0))} + \sum_{q(x) \in I} 4^{-(l(q(X))) + 1} = \frac{1 + \iota(I)}{4} \).

Approximation of \( f_P \) to accuracy \( \varepsilon \) yields

\[
\tilde{f}(x) \in \left[ \frac{1 + x}{4} - \varepsilon , \frac{1 + x}{4} + \varepsilon \right].
\]

Starting with some \( x \) and iterating \( \tilde{f} \) yields in the limit a value

\[
r \in \left[ \frac{1 - 4\varepsilon}{3} , \frac{1 + 4\varepsilon}{3} \right].
\]

Applying \( \iota^{-1} \) to \( r \) we find

\[
q(s^n(0)) \in \iota^{-1}(r) \text{ if } n < -\log_4 \varepsilon - 1.
\]
Approximation of Interpretations

- Let $P$ be a logic program over a first order language $\mathcal{L}$ and $l$ a level mapping.
- An interpretation $I$ approximates an interpretation $J$ to a degree $n \in \mathbb{N}$ if for all atoms $A \in B_\mathcal{L}$ with $l(A) < n$ we find $I(A) = t$ iff $J(A) = t$.

- $I$ approximates $J$ to a degree $n$ iff $d_\mathcal{L}(I, J) \leq 2^{-n}$. 
Approximation of Supported Models

- Given an acyclic logic program $P$ with bijective level mapping.
- Let $T_P$ be the immediate consequence operator associated with $P$ and $M_P$ the least supported model of $P$.
- We can approximate $T_P$ by a 3-layer feed forward network.
- We can turn this network into a recurrent one.

Does the recurrent network approximate the supported model of $P$?

**Theorem**  For an arbitrary $m \in \mathbb{N}$ there exists a recursive network with sigmoidal activation function for the hidden layer units and linear activation functions for the input and output layer units computing a function $\tilde{f}_P$ such that there exists an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ and for all $x \in [-1, 1]$ we find

$$d_L(\iota^{-1}(\tilde{f}_P^n(x)), M_P) \leq 2^{-m}.$$
Extensions

▶ Detailed study in (topological) continuity of semantic operators [Hitzler&Seda:2003], [Hitzler&H&Seda:2004]:
  ▶ many-valued logics,
  ▶ larger class of logic programs,
  ▶ other approximation theorems.

▶ A recursive network for reflexive reasoning [H&Kalinke&Wunderlich:2000].
▶ The graph of $f_P$ is an attractor of some iterated function system [Bader:2003], [Bader&Hitzler:2004]:
  ▶ representation theorems,
  ▶ fractal interpolation,
  ▶ recurrent radial basis function networks.

▶ Fibring Artificial Neural Networks [Bader&Garcez&Hitzler:2004].
Challenge Problem 1
How can first order terms be represented and manipulated in a connectionist system?

- Structured connectionist networks.
  - Quadratic number of units and cubic number of connections.

- Vectors of fixed length like RAAM, LRAAM or HRR.
  - Could not safely store and recall terms of depth larger than five.

- Hybrid systems
  - No real integration.

- Phase coding
  - Works only for data logic.

Idea  Connectionist encodings of conventional data structures.
Challenge Problem 2
How can first order rules be extracted from a connectionist system?

- Backpropagation can be used to train the kernels.
  - All known rule extraction techniques generate propositional rules.

Idea  Apply techniques from inductive logic programming.
Challenge Problem 3
How can multiple instances of first order rules be represented in a connectionist system?

- Current connectionist systems fix the number of copies beforehand.
  - The systems are incomplete.

Idea
- Use growing cell structures.
- Relate the accuracy of an approximation to available hardware resources.
Challenge Problem 4
What does a theory for the integration of logic and connectionist systems look like?

- Isolated systems.
- There is no theory so far.

Idea

- Various layers of increased expressiveness
- with corresponding connectionist models,
- their time and space complexities,
- their properties concerning learning and rule extraction and
- algorithms for learning and rule extraction.
Challenge Problem 5
Can such a theory be applied in real domains outperforming conventional approaches?

- All known applications of connectionist inference systems are toy examples.
- Parallel hardware is needed.
- Layered architecture.

Idea  Cognitive Robotics.
- Lower layers handle problems which are parallelizable.
- Upper layers handle deliberate reasoning.
- Life-long learning.
- Rule extraction.
Many thanks to Yvonne McIntyre, Hans-Peter Störr, Jörg Wunderlich, Pascal Hitzler and Sebastian Bader.

"Logic is everywhere ..."