Neural-Symbolic Integration

- Motivation
- Knowledge Based Artificial Neural Networks
- Core Method
- Propositional Core Method
- First-Order Core Method
- Discussion
Motivation: The Neural-Symbolic Cycle

Knowledge based artificial neural networks:

- networks are not arbitrarily initialized,
- but by the available background knowledge.
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

\[ \mathcal{P} = \{a \leftarrow b \land c \land \neg d, \quad a \leftarrow d \land \neg e, \quad h \leftarrow f \land g, \quad k \leftarrow a, \neg h\}. \]
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

$$\mathcal{P} = \{ a \leftarrow b \land c \land \neg d, \ a \leftarrow d \land \neg e, \ h \leftarrow f \land g, \ k \leftarrow a, \neg h \}.$$
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

$$\mathcal{P} = \{ a \leftarrow b \land c \land \neg d, \ a \leftarrow d \land \neg e, \ h \leftarrow f \land g, \ k \leftarrow a, \neg h \}. $$
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

\[ \mathcal{P} = \{ a \leftarrow b \land c \land \neg d, \ a \leftarrow d \land \neg e, \ h \leftarrow f \land g, \ k \leftarrow a, \neg h \}. \]
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

\[ \mathcal{P} = \{ a \leftarrow b \land c \land \neg d, \ a \leftarrow d \land \neg e, \ h \leftarrow f \land g, \ k \leftarrow a, \neg h \}. \]
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

$$\mathcal{P} = \{a \leftarrow b \land c \land \neg d, a \leftarrow d \land \neg e, h \leftarrow f \land g, k \leftarrow a, \neg h\}.$$
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

\[ \mathcal{P} = \{ a \leftarrow b \land c \land \neg d, \ a \leftarrow d \land \neg e, \ h \leftarrow f \land g, \ k \leftarrow a, \neg h \}. \]
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

\[ \mathcal{P} = \{ a \leftarrow b \land c \land \neg d, \ a \leftarrow d \land \neg e, \ h \leftarrow f \land g, \ k \leftarrow a, \neg h \}. \]
Knowledge Based Artificial Neural Networks

- Towell, Shavlik 1994: Can we do better than empirical learning?
- Consider hierarchical logic programs, e.g.,

\[ \mathcal{P} = \{a \leftarrow b \land c \land \neg d, a \leftarrow d \land \neg e, h \leftarrow f \land g, k \leftarrow a, \neg h\}. \]
Knowledge Based Artificial Neural Networks – Learning

- Given hierarchical sets of propositional rules as background knowledge.
- Map rules into multi-layer feed forward networks with sigmoidal units.
- Add hidden units (optional).
- Add units for known input features that are not referenced in the rules.
- Fully connect layers.
- Add near-zero random numbers to all links and thresholds.
- Apply backpropagation.

- Empirical evaluation: system performs better than purely empirical and purely hand-built classifiers.
Knowledge Based Artificial Neural Networks – A Problem

- Works if rules have few conditions and there are few rules with the same head.
Knowledge Based Artificial Neural Networks – A Problem

- Works if rules have few conditions and there are few rules with the same head.

![Diagram of neural network with nodes and connections]
Knowledge Based Artificial Neural Networks – A Problem

- Works if rules have few conditions and there are few rules with the same head.
Knowledge Based Artificial Neural Networks – A Problem

- Works if rules have few conditions and there are few rules with the same head.
Knowledge Based Artificial Neural Networks – A Problem

- Works if rules have few conditions and there are few rules with the same head.

![Diagram showing a network with nodes labeled q, q9, q10, r1, r2, r10, and edges connecting them. The network has a central node labeled s with subnodes 19ω/2, q, 19ω/2, and r, with further branching connecting q9 to q1 and q10, and r10 to r2.]
Knowledge Based Artificial Neural Networks – A Problem

- Works if rules have few conditions and there are few rules with the same head.

- $p_q = p_r = 9\omega$
Knowledge Based Artificial Neural Networks – A Problem

- Works if rules have few conditions and there are few rules with the same head.

\[ p_q = p_r = 9\omega \quad \text{and} \quad v_q = v_r = \frac{1}{1 + e^{\beta(9.5\omega - 9\omega)}} \approx 0.46 \quad \text{with} \quad \beta = 1. \]
Knowledge Based Artificial Neural Networks – A Problem

- Works if rules have few conditions and there are few rules with the same head.

- $p_q = p_r = 9\omega$ and $v_q = v_r = \frac{1}{1+e^{\beta(9.5\omega-9\omega)}} \approx 0.46$ with $\beta = 1$.

- $p_s = 0.92\omega$
Knowledge Based Artificial Neural Networks – A Problem

Works if rules have few conditions and there are few rules with the same head.

\[ p_q = p_r = 9\omega \quad \text{and} \quad v_q = v_r = \frac{1}{1 + e^{\beta(9.5\omega - 9\omega)}} \approx 0.46 \quad \text{with} \quad \beta = 1. \]

\[ p_s = 0.92\omega \quad \text{and} \quad v_s = \frac{1}{1 + e^{\beta(0.5\omega - 0.92\omega)}} \approx 0.6 \quad \text{with} \quad \beta = 1. \]
Knowledge Based Artificial Neural Networks – A Problem

Works if rules have few conditions and there are few rules with the same head.

\[ p_q = p_r = 9\omega \quad \text{and} \quad v_q = v_r = \frac{1}{1+e^{\beta(9.5\omega-9\omega)}} \approx 0.46 \quad \text{with} \quad \beta = 1. \]

\[ p_s = 0.92\omega \quad \text{and} \quad v_s = \frac{1}{1+e^{\beta(0.5\omega-0.92\omega)}} \approx 0.6 \quad \text{with} \quad \beta = 1. \]
The Core Method: The Very Idea

- Various semantics for logic programs coincide with fixed points of associated immediate consequence operators (e.g., Apt, vanEmden 1982).
The Core Method: The Very Idea

- Various semantics for logic programs coincide with fixed points of associated immediate consequence operators (e.g., Apt, van Emden 1982).

- **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 $y \in X$.

  - Fitting 1994: Consider logic programs, whose immediate consequence operator is a contraction.
The Core Method: The Very Idea

- Various semantics for logic programs coincide with fixed points of associated immediate consequence operators (e.g., Apt, vanEmden 1982).

- **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 $y \in X$.

  - Fitting 1994: Consider logic programs, whose immediate consequence operator is a contraction.

- Funahashi 1989: Every continuous function on the reals can be uniformly approximated by feedforward connectionist networks.

  - Hölldobler, Kalinke, Störr 1999: Consider logic programs, whose immediate consequence operator is continuous on the reals.
The Core Method: Blueprint of a Recurrent Network

- Given a logic program $\mathcal{P}$.
- Compute $T_\mathcal{P}$.
- Construct a feed-forward network – called core – computing or approximating $T_\mathcal{P}$.
- Turn the network into a recurrent one by connecting the output to the input layer.
Propositional Logic: Representing Interpretations

- Let $\mathcal{V}$ be a set of variables.
- $\mathcal{I} = 2^\mathcal{V}$
- Let $n = |\mathcal{V}|$ and identify $\mathcal{V}$ with $\{1, \ldots, n\}$.
- Define $\iota : \mathcal{I} \rightarrow \mathbb{R}^n$ such that for all $1 \leq j \leq n$ we find:
  \[
  \iota(I)[j] = \begin{cases} 
  1 & \text{if } j \in I, \\
  0 & \text{if } j \not\in I.
  \end{cases}
  \]
  
  E.g., if $\mathcal{V} = \{p, q, r\} = \{1, 2, 3\}$ and $I = \{p, r\}$ then $\iota(I) = (1, 0, 1)$.
- Other encodings are possible, e.g.,
  \[
  \iota(I)[j] = \begin{cases} 
  1 & \text{if } j \in I, \\
  -1 & \text{if } j \not\in I.
  \end{cases}
  \]
Propositional Core Method

- Consider the program \( \mathcal{P} = \{ p, r \leftarrow p \wedge \neg q, r \leftarrow \neg p \wedge q \} \).
- A translation algorithm translates \( \mathcal{P}_1 \) into a core of binary threshold units:
Propositional Core Method

- Consider the program $\mathcal{P} = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$.
- A translation algorithm translates $\mathcal{P}_1$ into a core of binary threshold units:
Propositional Core Method

Consider the program $\mathcal{P} = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$.

A translation algorithm translates $\mathcal{P}_1$ into a core of binary threshold units:
Propositional Core Method

- Consider the program $\mathcal{P} = \{p, \ r \leftarrow p \land \neg q, \ r \leftarrow \neg p \land q\}$.  
- A translation algorithm translates $\mathcal{P}_1$ into a core of binary threshold units:
Some Results

- **Proposition** 2-layer networks cannot compute $T_\mathcal{P}$ for definite $\mathcal{P}$.
- **Theorem** For each program $\mathcal{P}$, there exists a core computing $T_\mathcal{P}$.
- **Recall** $\mathcal{P} = \{ p, \; r \leftarrow p \land \neg q, \; r \leftarrow \neg p \land q \}$.
- **Adding recurrent connections:**
Some Results

- **Proposition** 2-layer networks cannot compute $T_\mathcal{P}$ for definite $\mathcal{P}$.
- **Theorem** For each program $\mathcal{P}$, there exists a core computing $T_\mathcal{P}$.
- **Recall** $\mathcal{P} = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$.
- **Adding recurrent connections:**

![Diagram showing recurrent connections](image-url)
Some Results

- **Proposition** 2-layer networks cannot compute $T_P$ for definite $P$.
- **Theorem** For each program $P$, there exists a core computing $T_P$.
- **Recall** $P = \{ p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q \}$.
- **Adding recurrent connections:**
Some Results

- **Proposition** 2-layer networks cannot compute $T_P$ for definite $P$.
- **Theorem** For each program $P$, there exists a core computing $T_P$.
- **Recall** $P = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$.
- **Adding recurrent connections:**
Some Results

- **Proposition** 2-layer networks cannot compute $T_P$ for definite $P$.
- **Theorem** For each program $P$, there exists a core computing $T_P$.
- **Recall** $P = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$.
- **Adding recurrent connections:**
Some Results

- **Proposition** 2-layer networks cannot compute $T_P$ for definite $P$.
- **Theorem** For each program $P$, there exists a core computing $T_P$.
- **Recall** $P = \{ p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q \}$.
- **Adding recurrent connections:**
Some Results

- **Proposition** 2-layer networks cannot compute $T_P$ for definite $P$.
- **Theorem** For each program $P$, there exists a core computing $T_P$.
- **Recall** $P = \{p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q\}$.
- **Adding recurrent connections:**
Some Results

- **Proposition** 2-layer networks cannot compute \( T_\mathcal{P} \) for definite \( \mathcal{P} \).
- **Theorem** For each program \( \mathcal{P} \), there exists a core computing \( T_\mathcal{P} \).
- **Recall** \( \mathcal{P} = \{ p, r \leftarrow p \land \neg q, r \leftarrow \neg p \land q \} \).
- **Adding recurrent connections:**
Strongly Determined Programs

A logic program $\mathcal{P}$ is said to be strongly determined if there exists a metric $d$ on the set of all Herbrand interpretations for $\mathcal{P}$ such that $T_\mathcal{P}$ is a contraction wrt $d$.

**Corollary** Let $\mathcal{P}$ be a strongly determined program. Then there exists a core with recurrent connections such that the computation with an arbitrary initial input converges and yields the unique fixed point of $T_\mathcal{P}$.

For more details see Hölldobler, Kalinke 94; Hitzler, Hölldobler, Seda 2004.
Propositional Core Method using Bipolar Sigmoidal Units

▶ d’Avila Garcez, Zaverucha, Carvalho 1997:
  Can we combine the ideas in Hölldobler, Kalinke 1994 and Towell, Shavlik 1994 while avoiding the problem with Towell’s and Shavlik’s approach?

▶ Consider propositional logic language.

Let $I$ be an interpretation and $a \in [0, 1]$.

$$\iota(I)[j] = \begin{cases} v \in [a, 1] & \text{if } j \in I, \\ w \in [-1, -a] & \text{if } j \not\in I. \end{cases}$$

▶ Replace threshold and sigmoidal units by bipolar sigmoidal ones, i.e., units with

$$\Phi(\vec{v}_k) = \theta_k = \sum_{j=1}^{m} w_{kj} v_j,$$
$$\Psi(\theta_k) = v_k = \frac{2}{1 + e^{\beta(\theta_k - \theta_k)}} - 1,$$

where $\theta_k \in \mathbb{R}$ is a threshold (or bias) and $\beta > 0$ a steepness parameter.
The Task

▶ How should $\alpha$, $\omega$ and $\theta_i$ be selected such that:

$\triangledown \ \nu_i \in [\alpha, 1]$ or $\nu_i \in [-1, -\alpha]$ and

$\triangledown$ the core computes the immediate consequence operator?

Hidden Layer Units

- Consider \( A \leftarrow L_1 \land \ldots \land L_n \).
- Let \( u \) be the hidden layer unit for this rule.
  - Suppose \( I \models L_1 \land \ldots \land L_n \).
    - \( u \) receives input \( \geq \omega a \) from unit representing \( L_i \).
    - \( p_u \geq n\omega a = p_u^+ \).
Hidden Layer Units

- Consider $A \leftarrow L_1 \land \ldots \land L_n$.
- Let $u$ be the hidden layer unit for this rule.
  - Suppose $I \models L_1 \land \ldots \land L_n$.
    - $u$ receives input $\geq \omega a$ from unit representing $L_i$.
    - $p_u \geq n\omega a = p_u^+$.
  - Suppose $I \not\models L_1 \land \ldots \land L_n$.
    - $u$ receives input $\leq -\omega a$ from at least one unit representing $L_i$.
    - $p_u \leq (n-1)\omega 1 - \omega a = p_u^-$.
Hidden Layer Units

Consider $A \leftarrow L_1 \land \ldots \land L_n$.

Let $u$ be the hidden layer unit for this rule.

- Suppose $I \models L_1 \land \ldots \land L_n$.
  - $u$ receives input $\geq \omega a$ from unit representing $L_i$.
  - $p_u \geq n \omega a = p_u^+$.

- Suppose $I \nvDash L_1 \land \ldots \land L_n$.
  - $u$ receives input $\leq -\omega a$ from at least one unit representing $L_i$.
  - $p_u \leq (n - 1) \omega 1 - \omega a = p_u^-$.

$\theta_u = \frac{n \omega a + (n-1)\omega - \omega a}{2} = (na + n - 1 - a)\frac{\omega}{2} = (n - 1)(a + 1)\frac{\omega}{2}$. 
Output Layer Units

- Let $\mu$ be the number of clause with head $A$.
- Consider $A \leftarrow L_1 \land \ldots \land L_n$.
- Suppose $I \models L_1 \land \ldots \land L_n$.

\[ p_A \geq \omega a + (\mu - 1)\omega(-1) = \omega a - (\mu - 1)\omega = p_A^+. \]
Output Layer Units

- Let $\mu$ be the number of clause with head $A$.
- Consider $A \leftarrow L_1 \land \ldots \land L_n$.
- Suppose $I \models L_1 \land \ldots \land L_n$.
  \[ p_A \geq \omega a + (\mu - 1)\omega(-1) = \omega a - (\mu - 1)\omega = p_A^+ \]
- Suppose for all rules of the form $A \leftarrow L_1 \land \ldots \land L_n$ we find $I \not\models L_1 \land \ldots \land L_n$.
  \[ p_A \leq -\mu\omega a = p_A^- \]
Output Layer Units

- Let \( \mu \) be the number of clause with head \( A \).
- Consider \( A \leftarrow L_1 \land \ldots \land L_n \).
- Suppose \( I \models L_1 \land \ldots \land L_n \).
  \[ p_A \geq \omega a + (\mu - 1)\omega(-1) = \omega a - (\mu - 1)\omega = p_A^+ \]
- Suppose for all rules of the form \( A \leftarrow L_1 \land \ldots \land L_n \) we find \( I \not\models L_1 \land \ldots \land L_n \).
  \[ p_A \leq -\mu \omega a = p_A^- \]
- \( \theta_A = \frac{\omega a - (\mu - 1)\omega - \mu a}{2} = (a - \mu + 1 - \mu a) \frac{\omega}{2} = (1 - \mu)(a + 1) \frac{\omega}{2} \).
Computing a Value for $a$

$\nabla \ p_u^+ > p_u^-$:

$\nabla \ n\omega a > (n - 1)\omega - \omega a.$
Computing a Value for $\alpha$

- $P_u^+ > P_u^-$:
  - $n\omega a > (n - 1)\omega - \omega a$.
  - $n\omega a + \omega a > (n - 1)\omega$. 
Computing a Value for $a$

$\boldsymbol{p_u^+ > p_u^-}$:

$\triangleright n\omega a > (n - 1)\omega - \omega a.$

$\triangleright n\omega a + \omega a > (n - 1)\omega.$

$\triangleright a(n + 1)\omega > (n - 1)\omega.$
Computing a Value for $a$

- $p^+_u > p^-_u$: 
  - $n\omega a > (n - 1)\omega - \omega a$. 
  - $n\omega a + \omega a > (n - 1)\omega$. 
  - $a(n + 1)\omega > (n - 1)\omega$. 
  - $a > \frac{n-1}{n+1}$. 
Computing a Value for $a$

$\boldsymbol{p_u^+ > p_u^-}$:

$\triangleright n\omega a > (n - 1)\omega - \omega a$.

$\triangleright n\omega a + \omega a > (n - 1)\omega$.

$\triangleright a(n + 1)\omega > (n - 1)\omega$.

$\triangleright a > \frac{n - 1}{n + 1}$.

$\boldsymbol{p_A^+ > p_A^-}$:

$\triangleright \omega a - (\mu - 1)\omega > -\mu a\omega$. 
Computing a Value for $a$

- $p_u^+ > p_u^-$:
  - $n\omega a > (n - 1)\omega - \omega a$.
  - $n\omega a + \omega a > (n - 1)\omega$.
  - $a(n + 1)\omega > (n - 1)\omega$.
  - $a > \frac{n-1}{n+1}$.

- $p_A^+ > p_A^-$:
  - $\omega a - (\mu - 1)\omega > -\mu a \omega$.
  - $\omega a + \mu a \omega > (\mu - 1)\omega$. 
Computing a Value for $a$

$\uparrow \quad p_u^+ > p_u^-:\n\quad n\omega a > (n - 1)\omega - \omega a.$
$\quad n\omega a + \omega a > (n - 1)\omega.$
$\quad a(n + 1)\omega > (n - 1)\omega.$
$\quad a > \frac{n-1}{n+1}.$

$\uparrow \quad p_A^+ > p_A^-:\n\quad \omega a - (\mu - 1)\omega > -\mu a\omega.$
$\quad \omega a + \mu a\omega > (\mu - 1)\omega.$
$\quad a(1 + \mu)\omega > (\mu - 1)\omega.$
Computing a Value for $a$

$\n \n\begin{align*}
\n\n\Rightarrow p_u^+ &> p_u^-: \\
\Rightarrow n\omega a &> (n - 1)\omega - \omega a. \\
\Rightarrow n\omega a + \omega a &> (n - 1)\omega. \\
\Rightarrow a(n + 1)\omega &> (n - 1)\omega. \\
\Rightarrow a &> \frac{n-1}{n+1}. \\
\n\Rightarrow p_A^+ &> p_A^-: \\
\Rightarrow \omega a - (\mu - 1)\omega &> -\mu a\omega. \\
\Rightarrow \omega a + \mu a\omega &> (\mu - 1)\omega. \\
\Rightarrow a(1 + \mu)\omega &> (\mu - 1)\omega. \\
\Rightarrow a &> \frac{\mu-1}{\mu+1}. \\
\end{align*}
\n\n
Computing a Value for $a$

- $p^+_u > p^-_u$:
  - $n\omega a > (n - 1)\omega - \omega a$.
  - $n\omega a + \omega a > (n - 1)\omega$.
  - $a(n + 1)\omega > (n - 1)\omega$.
  - $a > \frac{n - 1}{n + 1}$.

- $p^+_A > p^-_A$:
  - $\omega a - (\mu - 1)\omega > -\mu a\omega$.
  - $\omega a + \mu a\omega > (\mu - 1)\omega$.
  - $a(1 + \mu)\omega > (\mu - 1)\omega$.
  - $a > \frac{\mu - 1}{\mu + 1}$.

- Consider all rules $\Rightarrow$ minimum value for $a$. 
Computing a Value for $\omega$

$\Psi(p) = \frac{2}{1 + e^{\beta(\theta - p)}} - 1 \geq a.$
Computing a Value for $\omega$

- $\Psi(p) = \frac{2}{1+e^{\beta(\theta-p)}} - 1 \geq \alpha.$
- $\frac{2}{1+e^{\beta(\theta-p)}} \geq 1 + \alpha.$
Computing a Value for $\omega$

- $\Psi(p) = \frac{2}{1 + e^{\beta(\theta - p)}} - 1 \geq a$.
- $\frac{2}{1 + e^{\beta(\theta - p)}} \geq 1 + a$.
- $\frac{2}{1 + a} \geq 1 + e^{\beta(\theta - p)}$. 
Computing a Value for $\omega$

$\Psi(p) = \frac{2}{1 + e^{\beta(\theta - p)}} - 1 \geq a.$

$\frac{2}{1 + e^{\beta(\theta - p)}} \geq 1 + a.$

$\frac{2}{1 + a} \geq 1 + e^{\beta(\theta - p)}.$

$\frac{2}{1 + a} - 1 = \frac{2}{1 + a} - \frac{1 + a}{1 + a} = \frac{1 - a}{1 + a} \geq e^{\beta(\theta - p)}.$
Computing a Value for $\omega$

- $\Psi(p) = \frac{2}{1 + e^{\beta(\theta - p)}} - 1 \geq a$.
- $\frac{2}{1 + e^{\beta(\theta - p)}} \geq 1 + a$.
- $\frac{2}{1 + a} \geq 1 + e^{\beta(\theta - p)}$.
- $\frac{2}{1 + a} - 1 = \frac{2}{1 + a} - \frac{1 + a}{1 + a} = \frac{1 - a}{1 + a} \geq e^{\beta(\theta - p)}$.
- $\ln(\frac{1 - a}{1 + a}) \geq \beta(\theta - p)$. 
Computing a Value for $\omega$

1. $\Psi(p) = \frac{2}{1+e^{\beta(\theta-p)}} - 1 \geq a$.
2. $\frac{2}{1+e^{\beta(\theta-p)}} \geq 1 + a$.
3. $\frac{2}{1+a} \geq 1 + e^{\beta(\theta-p)}$.
4. $\frac{2}{1+a} - 1 = \frac{2}{1+a} - \frac{1+a}{1+a} = \frac{1-a}{1+a} \geq e^{\beta(\theta-p)}$.
5. $\ln\left(\frac{1-a}{1+a}\right) \geq \beta(\theta - p)$.
6. $\frac{1}{\beta} \ln\left(\frac{1-a}{1+a}\right) \geq \theta - p$. 
Computing a Value for $\omega$

- $\Psi(p) = \frac{2}{1+e^{\beta(\theta-p)}} - 1 \geq a$.
- $\frac{2}{1+e^{\beta(\theta-p)}} \geq 1 + a$.
- $\frac{2}{1+a} \geq 1 + e^{\beta(\theta-p)}$.
- $\frac{2}{1+a} - 1 = \frac{2}{1+a} - \frac{1+a}{1+a} = \frac{1-a}{1+a} \geq e^{\beta(\theta-p)}$.
- $\ln(\frac{1-a}{1+a}) \geq \beta(\theta - p)$.
- $\frac{1}{\beta} \ln(\frac{1-a}{1+a}) \geq \theta - p$.

Consider a hidden layer unit:

- $\frac{1}{\beta} \ln(\frac{1-a}{1+a}) \geq (n - 1)(a + 1)^\omega - n\omega a = \frac{n(1-a) - 1 - 2na}{2}\omega = \frac{n - 1 - a(n+1)}{2}\omega$. 
Computing a Value for $\omega$

- $\Psi(p) = \frac{2}{1+e^{\beta(\theta-p)}} - 1 \geq a$.
- $\frac{2}{1+e^{\beta(\theta-p)}} \geq 1 + a$.
- $\frac{2}{1+a} \geq 1 + e^{\beta(\theta-p)}$.
- $\frac{2}{1+a} - 1 = \frac{2}{1+a} - \frac{1+a}{1+a} = \frac{1-a}{1+a} \geq e^{\beta(\theta-p)}$.
- $\ln(\frac{1-a}{1+a}) \geq \beta(\theta - p)$.
- $\frac{1}{\beta} \ln(\frac{1-a}{1+a}) \geq \theta - p$.

Consider a hidden layer unit:

- $\frac{1}{\beta} \ln(\frac{1-a}{1+a}) \geq (n - 1)(a + 1)\frac{\omega}{2} - n\omega a = \frac{na+n-a-1-2na}{2}\omega = \frac{n-1-a(n+1)}{2}\omega$.
- $\omega \geq \frac{2}{(n-1-a(n+1))\beta} \ln(\frac{1-a}{1+a})$ because $a \geq \frac{n-1}{n+1}$. 

Computing a Value for $\omega$

- $\Psi(p) = \frac{2}{1+e^{\beta(\theta - p)}} - 1 \geq a$.
- $\frac{2}{1+e^{\beta(\theta - p)}} \geq 1 + a$.
- $\frac{2}{1+a} \geq 1 + e^{\beta(\theta - p)}$.
- $\frac{2}{1+a} - 1 = \frac{2}{1+a} - \frac{1+a}{1+a} = \frac{1-a}{1+a} \geq e^{\beta(\theta - p)}$.
- $\ln\left(\frac{1-a}{1+a}\right) \geq \beta(\theta - p)$.
- $\frac{1}{\beta} \ln\left(\frac{1-a}{1+a}\right) \geq \theta - p$.

Consider a hidden layer unit:

- $\frac{1}{\beta} \ln\left(\frac{1-a}{1+a}\right) \geq (n-1)(a+1)\frac{\omega}{2} - n\omega a = \frac{na+n-a-1-2na}{2}\omega = \frac{n-1-a(n+1)}{2}\omega$.
- $\omega \geq \frac{2}{(n-1-a(n+1))\beta} \ln\left(\frac{1-a}{1+a}\right)$ because $a \geq \frac{n-1}{n+1}$.

Consider all hidden and output layer units as well as the case that $\Psi(p) \leq -a$:

$\rightsquigarrow$ minimum value for $\omega$. 
Results

- Relation to logic programs is preserved.
- The core is trainable by backpropagation.
- Many interesting applications, e.g.:
  - DNA sequence analysis.
  - Power system fault diagnosis.
- Empirical evaluation:
  system performs better than well-known machine learning systems.
- See d’Avila Garcez, Broda, Gabbay 2002 for details.
Extensions

- Many-valued logic programs
- Modal logic programs
- Intuitionistic logic programs
- Answer set programming
- Metalevel priorities
- Rule extraction
Literature

First-Order Core Method

- Hölldobler, Kalinke, Störr 1999:
  Can the core method be extended to first-order logic programs?

- Problem

  - Given a logic program $\mathcal{P}$ over a first order language $\mathcal{L}$
    together with $T_\mathcal{P} : 2^{B_\mathcal{L}} \rightarrow 2^{B_\mathcal{L}}$.
  - $B_\mathcal{L}$ 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 \( \iota : 2^{BL} \to \mathbb{R}^m \) and \( f_P : \mathbb{R}^m \to \mathbb{R}^m \) such that the following conditions hold.
  - \( T_P(I) = I' \) implies \( f_P(\iota(I)) = \iota(I') \).
  - \( f_P(\vec{x}) = \vec{x}' \) implies \( T_P(\iota^{-1}(\vec{x})) = \iota^{-1}(\vec{x}') \).

\( \implies f_P \) is a sound and complete encoding of \( T_P \).

- \( T_P \) is a contraction on \( 2^{BL} \) iff \( f_P \) is a contraction on \( \mathbb{R}^m \).
  \( \implies \) The contraction property and fixed points are preserved.

- \( f_P \) is continuous on \( \mathbb{R}^m \).
  \( \implies \) A connectionist network approximating \( f_P \) is known to exist.
  \( \implies f_P \) and, hence, \( T_P \) can be trained by backpropagation and related training methods.
Acyclic Logic Programs

- Let \( \mathcal{P} \) be a program over a first order language \( \mathcal{L} \).
- A level mapping for \( \mathcal{P} \) is a function \( l : B_\mathcal{L} \rightarrow \mathbb{N} \).
  - 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 (Fitting 1994) \( (2^{B_\mathcal{L}}, d_\mathcal{L}) \) is a complete metric space.
- \( \mathcal{P} \) is said to be acyclic wrt a level mapping \( l \),
  if for every \( A \leftarrow L_1 \wedge \ldots \wedge L_n \in \text{ground}(\mathcal{P}) \) we find \( l(A) > l(L_i) \) for all \( i \).
- Proposition Let \( \mathcal{P} \) be an acyclic logic program wrt \( l \) and \( d_\mathcal{L} \) the metric associated with \( \mathcal{L} \) and \( l \), then \( T_\mathcal{P} \) 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 $l$ be a bijective level mapping.
- $\{\top, \bot\}$ can be identified with $\{0, 1\}$.
- The set of all mappings $B_L \rightarrow \{\top, \bot\}$ can be identified with the set of all mappings $\mathbb{N} \times S \rightarrow \{0, 1\}$.
- Let $I_L$ be the set of all mappings from $B_L$ to $\{0, 1\}$.
- Let $R : I_L \rightarrow \mathcal{D}$ be defined as

$$
\nu(I) = \sum_{i=1}^{\infty} I(l^{-1}(i))4^{-i}.
$$

- Proposition $R$ is a bijection.

We have a sound and complete encoding of interpretations.
We define $f_\mathcal{P}: \mathcal{D} \to \mathcal{D} : r \mapsto \iota(T_\mathcal{P}(R^{-1}(r)))$.

We have a sound and complete encoding of $T_\mathcal{P}$.

**Proposition** Let $\mathcal{P}$ be an acyclic program wrt a bijective level mapping. $f_\mathcal{P}$ is a contraction on $\mathcal{D}$.

Contraction property and fixed points are preserved.
Approximating Continuous Functions

- **Corollary** \( f_P \) is continuous.

- **Recall Funahashi’s theorem:**
  - Every continuous function \( f : K \to \mathbb{R} \) 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 \( R^{-1} \).

  Connectionist network approximating immediate consequence operator exists.
An Example

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

- $\mathcal{P}$ is acyclic wrt $l$, $l$ is bijective, $\iota(B_L) = \frac{1}{3}$.

- $f_{\mathcal{P}}(\iota(I)) = 4^{-l(q(0))} + \sum_{q(X) \in I} 4^{-l(q(s(X)))} = 4^{-l(q(0))} + \sum_{q(X) \in I} 4^{-l(q(X))} = \frac{1 + \iota(I)}{4}$.

Approximation of $f_{\mathcal{P}}$ to accuracy $\varepsilon$ yields

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

Starting with some $x$ and iterating $f$ yields in the limit a value

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

Applying $R^{-1}$ to $r$ we find

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

- Let \( \mathcal{P} \) 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 \( I(A) = \top \) iff \( J(A) = \top \).

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

- Given an acyclic logic program $\mathcal{P}$ with bijective level mapping.
- Let $T_\mathcal{P}$ be the immediate consequence operator associated with $\mathcal{P}$ and $M_\mathcal{P}$ the least supported model of $\mathcal{P}$.
- We can approximate $T_\mathcal{P}$ 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 $\mathcal{P}$?

**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 $\bar{f}_\mathcal{P}$ 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

$$d_L(R^{-1}(\bar{f}_\mathcal{P}^n(x))), M_\mathcal{P}) \leq 2^{-m}.$$