The Algorithms of Unification and SLD Resolution in Neural Networks

Ekaterina Komendantskaya

INRIA Sophia Antipolis, France

A talk in Universities of St.Andrews, Bath, Imperial College
3 - 6 March 2008

Outline

1. Motivation
   - Neuro-Symbolic Integration
   - Connectionist Neural Networks and Logic Programs

Outline

1. Motivation
   - Neuro-Symbolic Integration
   - Connectionist Neural Networks and Logic Programs
2. SLD-resolution

Outline

1. Motivation
   - Neuro-Symbolic Integration
   - Connectionist Neural Networks and Logic Programs
2. SLD-resolution
3. First-Order Deduction in Neural networks

Outline

1. Motivation
   - Neuro-Symbolic Integration
   - Connectionist Neural Networks and Logic Programs
2. SLD-resolution
3. First-Order Deduction in Neural networks
4. Conclusions and Future Work

Motivation

Symbolic Logic as Deductive System

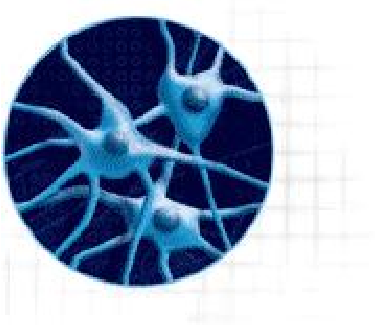
1. Axioms: $(A \supset (B \supset A))$;
$(A \supset (B \supset C)) \supset$
$((A \supset B) \supset (A \supset C))$;
$(\neg\neg A \supset A)$;
$((\forall x A) \supset S_t^x A)$;
$\forall x(A \supset B) \supset$
$(A \supset \forall x B)$;
2. Rules:
$\dfrac{A \supset B,\ A}{B}$; $\dfrac{A}{\forall x A}$.

Motivation

Symbolic Logic as Deductive System

1. Axioms: $(A \supset (B \supset A))$; $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$; $(\neg\neg A \supset A)$; $((\forall x A) \supset S^x_t A)$; $\forall x(A \supset B) \supset (A \supset \forall x B)$;
2. Rules: $\dfrac{A \supset B,\ A}{B}$; $\dfrac{A}{\forall x A}$.

Neural Networks



- spontaneous behavior;
- learning and adaptation

Motivation

Logic Programs

- $A \leftarrow B_1, \ldots, B_n$

Motivation

Logic Programs

- $A \leftarrow B_1, \ldots, B_n$
- $T_P(I) = \{A \in B_P :$
  $A \leftarrow B_1, \ldots, B_n$
  is a ground instance of a clause in P and
  $\{B_1, \ldots, B_n\} \subseteq I\}$

Motivation

Logic Programs

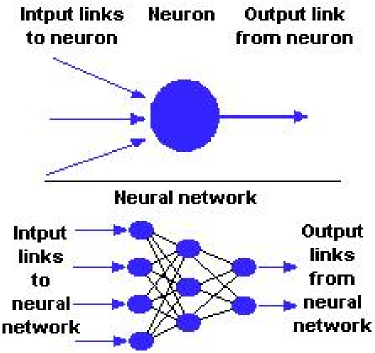
- $A \leftarrow B_1, \ldots, B_n$
- $T_P(I) = \{A \in B_P :$
  $A \leftarrow B_1, \ldots, B_n$
  is a ground instance of a clause in P and
  $\{B_1, \ldots, B_n\} \subseteq I\}$
- $\mathtt{lfp}(T_P \uparrow \omega) =$ the least Herbrand model of P.

Motivation

Logic Programs

- $A \leftarrow B_1, \ldots, B_n$
- $T_P(I) = \{A \in B_P :$ $A \leftarrow B_1, \ldots, B_n$ is a ground instance of a clause in P and $\{B_1, \ldots, B_n\} \subseteq I\}$
- $\texttt{lfp}(T_P \uparrow \omega) =$ the least Herbrand model of P.

Artificial Neural Networks



An Important Result, [Kalinke, Hölldobler, 94]

Theorem

For each propositional program P, there exists a 3-layer recurrent neural network which computes T_P.

An Important Result, [Kalinke, Hölldobler, 94]

Theorem

For each propositional program P, there exists a 3-layer recurrent neural network which computes T_P.

- No learning or adaptation;
- First-order atoms are not represented in the neural networks directly, and only truth values 0 and 1 are propagated.
- Require infinitely long layers in the first-order case.

A Simple Example

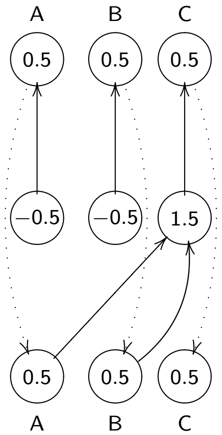
$B \leftarrow$
$A \leftarrow$
$C \leftarrow A, B$

$T_P \uparrow 0 = \{B, A\}$
$lfp(T_P) = T_P \uparrow 1 = \{B, A, C\}$

A Simple Example

$B \leftarrow$
$A \leftarrow$
$C \leftarrow A, B$

$T_P \uparrow 0 = \{B, A\}$
$lfp(T_P) = T_P \uparrow 1 = \{B, A, C\}$



A Simple Example

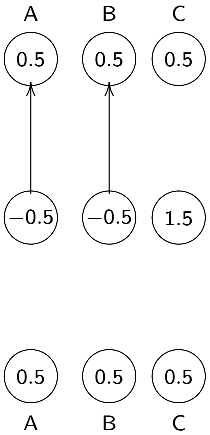
$B \leftarrow$

$A \leftarrow$

$C \leftarrow A, B$

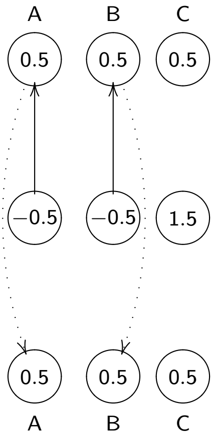
$T_P \uparrow 0 = \{B, A\}$

$lfp(T_P) = T_P \uparrow 1 = \{B, A, C\}$



A Simple Example

$B \leftarrow$
$A \leftarrow$
$C \leftarrow A, B$
$T_P \uparrow 0 = \{B, A\}$
$lfp(T_P) = T_P \uparrow 1 = \{B, A, C\}$



A Simple Example

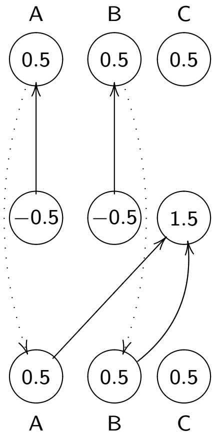
$B \leftarrow$

$A \leftarrow$

$C \leftarrow A, B$

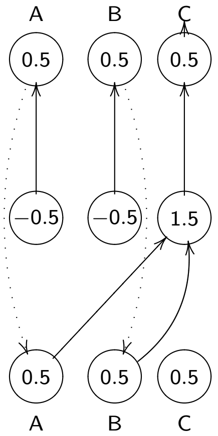
$T_P \uparrow 0 = \{B, A\}$

$lfp(T_P) = T_P \uparrow 1 = \{B, A, C\}$



A Simple Example

$B \leftarrow$
$A \leftarrow$
$C \leftarrow A, B$
$T_P \uparrow 0 = \{B, A\}$
$lfp(T_P) = T_P \uparrow 1 = \{B, A, C\}$



Another Example: First-Order Case

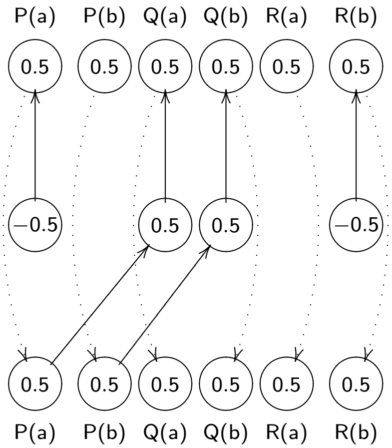
$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$

$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 =$
$\{P(a), R(b), Q(a)\}$

Another Example: First-Order Case

$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$

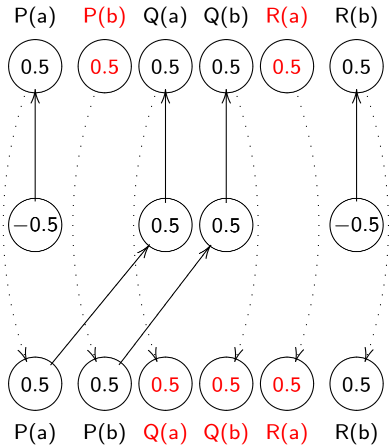
$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 =$
$\{P(a), R(b), Q(a)\}$



Another Example: First-Order Case

$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$

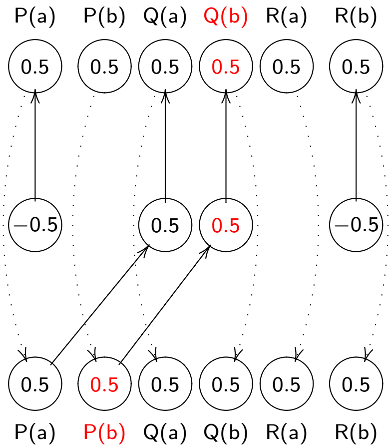
$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 =$
$\{P(a), R(b), Q(a)\}$



Another Example: First-Order Case

$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$

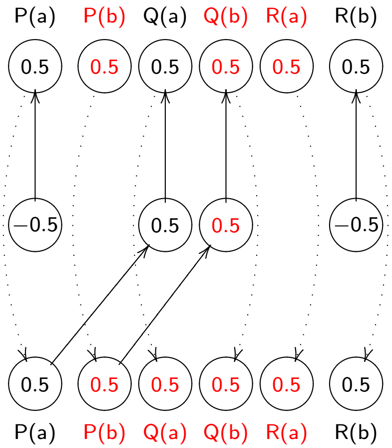
$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 =$
$\{P(a), R(b), Q(a)\}$



Another Example: First-Order Case

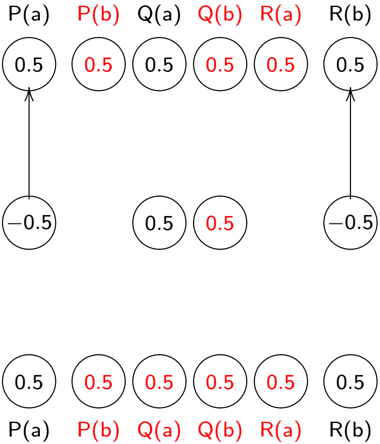
$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$

$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 = \{P(a), R(b), Q(a)\}$



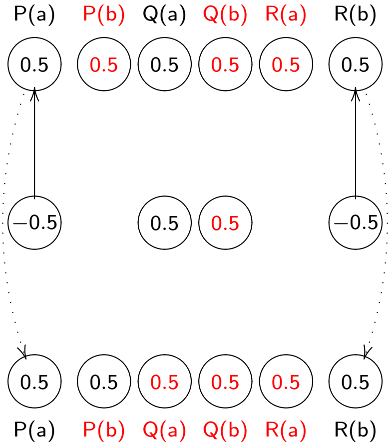
Another Example: First-Order Case

$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$
$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 =$
$\{P(a), R(b), Q(a)\}$



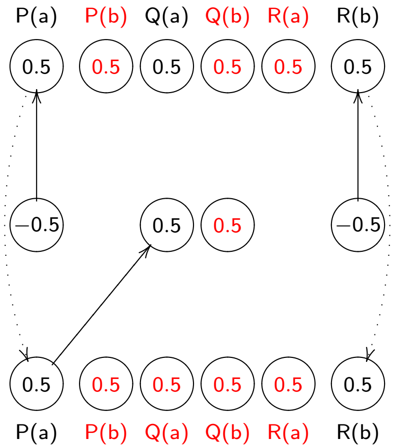
Another Example: First-Order Case

$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$
$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 = \{P(a), R(b), Q(a)\}$



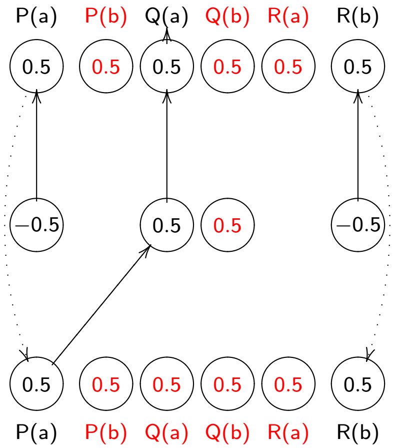
Another Example: First-Order Case

$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$
$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 = \{P(a), R(b), Q(a)\}$



Another Example: First-Order Case

$P(a) \leftarrow$
$Q(x) \leftarrow P(x)$
$R(b) \leftarrow$
$T_P \uparrow 0 = \{P(a), R(b)\}$
$lfp(T_P) = T_P \uparrow 1 =$
$\{P(a), R(b), Q(a)\}$



Example 3

$P(0) \leftarrow$
$P(s(x)) \leftarrow P(x)$

$T_P \uparrow 0 = \{P(0)\}$
$lfp(T_P) = T_P \uparrow \omega =$
$\{0, s(0), s(s(0)),$
$s(s(s(0))), \ldots\}$

Example 3

$P(0) \leftarrow$
$P(s(x)) \leftarrow P(x)$

$T_P \uparrow 0 = \{P(0)\}$
$lfp(T_P) = T_P \uparrow \omega =$
$\{0, s(0), s(s(0)),$
$s(s(s(0))), \ldots\}$

Example 3

$P(0) \leftarrow$
$P(s(x)) \leftarrow P(x)$

$T_P \uparrow 0 = \{P(0)\}$
$lfp(T_P) = T_P \uparrow \omega =$
$\{0, s(0), s(s(0)),$
$s(s(s(0))), \ldots\}$

Paradox:
(computability,
complexity,
proof theory)

Most General Unifier

MGU

Let S be a finite set of atoms. A substitution θ is called a unifier for S if S is a singleton. A unifier θ for S is called a *most general unifier* (mgu) for S if, for each unifier σ of S, there exists a substitution γ such that $\sigma = \theta\gamma$.

Example: If $S = (P(x), P(0))$, then $\theta = \{x/0\}$ is the mgu.

Disagreement set

Disagreement set

To find the *disagreement set* D_S of S locate the leftmost symbol position at which not all atoms in S have the same symbol and extract from each atom in S the term beginning at that symbol position. The set of all such terms is the disagreement set.

Example: For $S = (Q(f(x, y)), Q(f(a, b)))$ we have $D_S = \{x,\ a\}$.

Unification algorithm

1. Put $k = 0$ and $\sigma_0 = \varepsilon$.
2. If $S\sigma_k$ is a singleton, then stop; σ_k is an mgu of S. Otherwise, find the disagreement set D_k of $S\sigma_k$.
3. If there exist a variable v and a term t in D_k such that v does not occur in t, then put $\theta_{k+1} = \theta_k\{v/t\}$, increment k and go to 2. Otherwise, stop; S is not unifiable.

Unification algorithm

1. Put $k = 0$ and $\sigma_0 = \varepsilon$.
2. If $S\sigma_k$ is a singleton, then stop; σ_k is an mgu of S. Otherwise, find the disagreement set D_k of $S\sigma_k$.
3. If there exist a variable v and a term t in D_k such that v does not occur in t, then put $\theta_{k+1} = \theta_k\{v/t\}$, increment k and go to 2. Otherwise, stop; S is not unifiable.

Unification theorem.

SLD-resolution - Example

$$\begin{aligned} P(0) &\leftarrow \\ P(s(x)) &\leftarrow P(x) \end{aligned}$$

SLD-resolution - Example

$$\begin{array}{rcl} P(0) & \leftarrow & \\ P(s(x)) & \leftarrow & P(x) \end{array}$$

1. $G_0 = \leftarrow P(x)$.

SLD-resolution - Example

$$\begin{array}{rcl} P(0) & \leftarrow & \\ P(s(x)) & \leftarrow & P(x) \end{array}$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$.

SLD-resolution - Example

$$\begin{array}{rcl} P(0) & \leftarrow & \\ P(s(x)) & \leftarrow & P(x) \end{array}$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$.

SLD-resolution - Example

$$
\begin{array}{rcl}
P(0) & \leftarrow & \\
P(s(x)) & \leftarrow & P(x)
\end{array}
$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$. $S\theta_1 = \{P(0)\}$ is a singleton.
   Answer: 0.

SLD-resolution - Example

$$P(0) \leftarrow$$
$$P(s(x)) \leftarrow P(x)$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$. $S\theta_1 = \{P(0)\}$ is a singleton.
   Answer: 0.
2. $G_0 = \leftarrow P(x)$.

SLD-resolution - Example

$$P(0) \leftarrow$$
$$P(s(x)) \leftarrow P(x)$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$. $S\theta_1 = \{P(0)\}$ is a singleton.
   Answer: 0.
2. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(s(x))\}$.

SLD-resolution - Example

$$P(0) \leftarrow$$
$$P(s(x)) \leftarrow P(x)$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$. $S\theta_1 = \{P(0)\}$ is a singleton.
   Answer: 0.
2. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(s(x))\}$. $D_S = \{x, s(x)\}$. Put $\theta_1 = x/s(x)$.

SLD-resolution - Example

$$\begin{array}{rcl} P(0) & \leftarrow & \\ P(s(x)) & \leftarrow & P(x) \end{array}$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$. $S\theta_1 = \{P(0)\}$ is a singleton.
   Answer: 0.
2. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(s(x))\}$. $D_S = \{x, s(x)\}$. Put $\theta_1 = x/s(x)$. $S\theta_1 = \{P(s(x))\}$ is a singleton.

SLD-resolution - Example

$$
\begin{aligned}
P(0) &\leftarrow \\
P(s(x)) &\leftarrow \quad P(x)
\end{aligned}
$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$. $S\theta_1 = \{P(0)\}$ is a singleton.
   Answer: 0.
2. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(s(x))\}$. $D_S = \{x, s(x)\}$. Put $\theta_1 = x/s(x)$. $S\theta_1 = \{P(s(x))\}$ is a singleton.
   $G_1 = \leftarrow P(s(x))$.

SLD-resolution - Example

$$
\begin{aligned}
P(0) &\leftarrow \\
P(s(x)) &\leftarrow P(x)
\end{aligned}
$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$. $S\theta_1 = \{P(0)\}$ is a singleton.
   Answer: 0.
2. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(s(x))\}$. $D_S = \{x, s(x)\}$. Put $\theta_1 = x/s(x)$. $S\theta_1 = \{P(s(x))\}$ is a singleton.
   $G_1 = \leftarrow P(s(x))$. $S = \{P(s(x)), P(s(x)\}$. $D_S = \{\emptyset\}$.

SLD-resolution - Example

$$P(0) \leftarrow$$
$$P(s(x)) \leftarrow P(x)$$

1. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(0)\}$. $D_S = \{x, 0\}$. Put $\theta_1 = x/0$. $S\theta_1 = \{P(0)\}$ is a singleton.
   Answer: 0.
2. $G_0 = \leftarrow P(x)$. $S = \{P(x), P(s(x))\}$. $D_S = \{x, s(x)\}$. Put $\theta_1 = x/s(x)$. $S\theta_1 = \{P(s(x))\}$ is a singleton.
   $G_1 = \leftarrow P(s(x))$. $S = \{P(s(x)), P(s(x)\}$. $D_S = \{\emptyset\}$.
   $G_2 = \leftarrow P(x)$; search can go on as in item 1 ($\theta_2 = x/0$, answer $s(0)$); or as in item 2 (answers $s(s(0)), \ldots$).

Gödel Numbers of Formulae

Each symbol of the first-order language receives a **Gödel number** as follows:

- variables $x_1, x_2, x_3, \ldots$ receive numbers $(01), (011), (0111), \ldots$;
- constants $a_1, a_2, a_3, \ldots$ receive numbers $(21), (211), (2111), \ldots$;
- function symbols $f_1, f_2, f_3, \ldots$ receive numbers $(31), (311), (3111), \ldots$;
- predicate symbols $Q_1, Q_2, Q_3, \ldots$ receive numbers $(41), (411), (4111), \ldots$;
- symbols (,) and , receive numbers 5, 6 and 7 respectively.

Operations on Gödel Numbers

- **Disagreement set**: $g_1 \ominus g_2$;

Operations on Gödel Numbers

- **Disagreement set**: $g_1 \ominus g_2$;
- **Concatenation**: $g_1 \oplus g_2$;

Operations on Gödel Numbers

- **Disagreement set**: $g_1 \ominus g_2$;
- **Concatenation**: $g_1 \oplus g_2$;
- **Gödel number of substitution**: $s(g_1, g_2)$;

Operations on Gödel Numbers

- **Disagreement set**: $g_1 \ominus g_2$;
- **Concatenation**: $g_1 \oplus g_2$;
- **Gödel number of substitution**: $s(g_1, g_2)$;
- **Applying the substitution**: $g \odot s$;

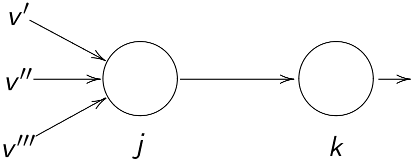
Operations on Gödel Numbers

- **Disagreement set**: $g_1 \ominus g_2$;
- **Concatenation**: $g_1 \oplus g_2$;
- **Gödel number of substitution**: $s(g_1, g_2)$;
- **Applying the substitution**: $g \odot s$;
- **Algorithm of unification**.

Neurons in Connectionist Neural Networks

$$p_k(t) = \left(\sum_{j=1}^{n_k} w_{kj} v_j(t)\right) - \Theta_k$$

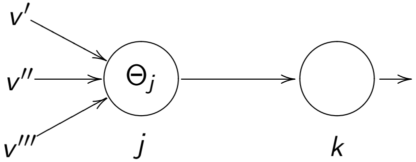
$$v_k(t + \Delta t) = \psi(p_k(t)) = \begin{cases} 1 & \texttt{if } \; p_k(t) > 0 \\ 0 & \texttt{otherwise.} \end{cases}$$



Neurons in Connectionist Neural Networks

$$p_k(t) = \left(\sum_{j=1}^{n_k} w_{kj} v_j(t)\right) - \Theta_k$$

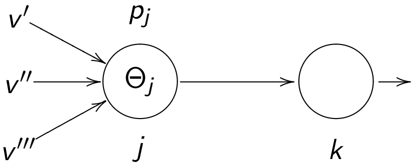
$$v_k(t + \Delta t) = \psi(p_k(t)) = \begin{cases} 1 & \texttt{if } \ p_k(t) > 0 \\ 0 & \texttt{otherwise.} \end{cases}$$



Neurons in Connectionist Neural Networks

$$p_k(t) = \left(\sum_{j=1}^{n_k} w_{kj} v_j(t)\right) - \Theta_k$$

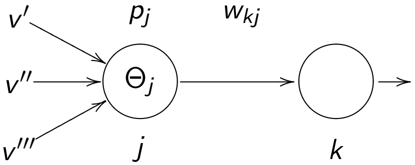
$$v_k(t + \Delta t) = \psi(p_k(t)) = \begin{cases} 1 & \texttt{if } p_k(t) > 0 \\ 0 & \texttt{otherwise.} \end{cases}$$



Neurons in Connectionist Neural Networks

$$p_k(t) = \left(\sum_{j=1}^{n_k} w_{kj} v_j(t)\right) - \Theta_k$$

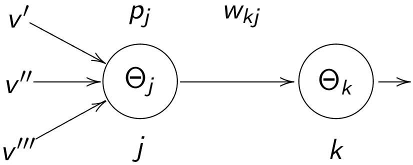
$$v_k(t + \Delta t) = \psi(p_k(t)) = \begin{cases} 1 & \texttt{if } \; p_k(t) > 0 \\ 0 & \texttt{otherwise.} \end{cases}$$



Neurons in Connectionist Neural Networks

$$p_k(t) = \left(\sum_{j=1}^{n_k} w_{kj} v_j(t)\right) - \Theta_k$$

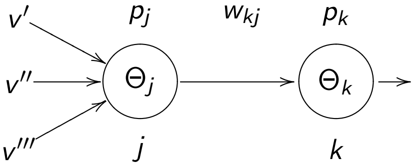
$$v_k(t + \Delta t) = \psi(p_k(t)) = \begin{cases} 1 & \texttt{if } \; p_k(t) > 0 \\ 0 & \texttt{otherwise.} \end{cases}$$



Neurons in Connectionist Neural Networks

$$p_k(t) = \left(\sum_{j=1}^{n_k} w_{kj} v_j(t)\right) - \Theta_k$$

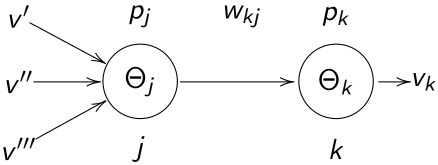
$$v_k(t + \Delta t) = \psi(p_k(t)) = \begin{cases} 1 & \texttt{if} \ \ p_k(t) > 0 \\ 0 & \texttt{otherwise.} \end{cases}$$



Neurons in Connectionist Neural Networks

$$p_k(t) = \left(\sum_{j=1}^{n_k} w_{kj} v_j(t)\right) - \Theta_k$$

$$v_k(t + \Delta t) = \psi(p_k(t)) = \begin{cases} 1 & \texttt{if } p_k(t) > 0 \\ 0 & \texttt{otherwise.} \end{cases}$$

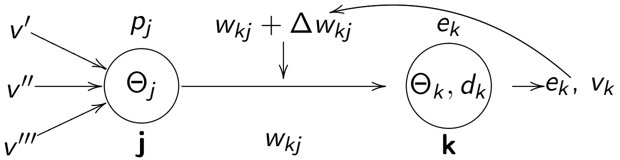


Unification in Neural Networks

Claim 1

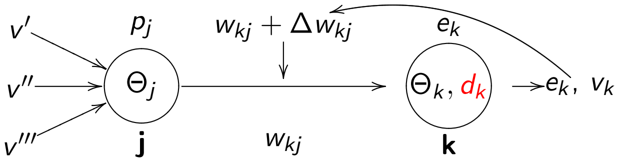
Unification Algorithm can be performed in finite (and very small) neural networks with error-correction learning.

Error-Correction (Supervised) Learning



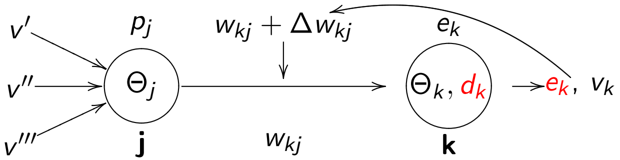
Error-Correction (Supervised) Learning

We embed a new parameter, **desired response** d_k into neurons;



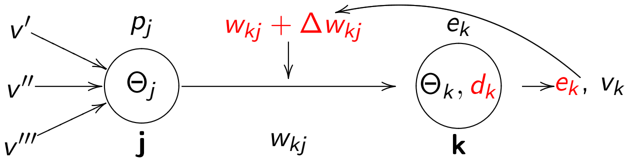
Error-Correction (Supervised) Learning

We embed a new parameter, **desired response** d_k into neurons;
Error-signal: $e_k(t) = d_k(t) - v_k(t)$;



Error-Correction (Supervised) Learning

We embed a new parameter, **desired response** d_k into neurons;
Error-signal: $e_k(t) = d_k(t) - v_k(t)$;
Error-correction learning rule: $\Delta w_{kj}(t) = \eta e_k(t) v_j(t)$.

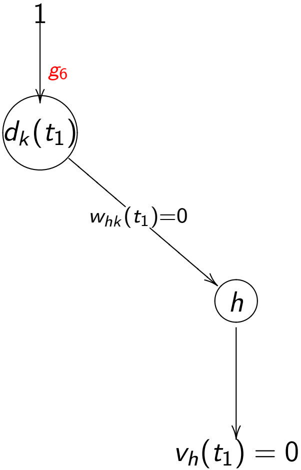


Main Lemma

Lemma

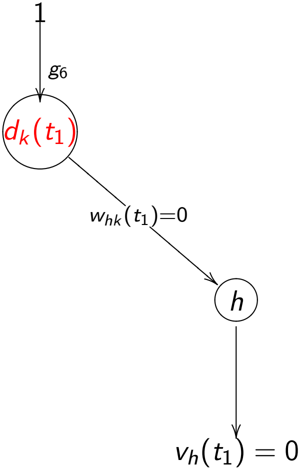
Given two first-order atoms A and B, there exists a two-neuron learning neural network that performs the algorithm of unification for A and B.

Example of Unification in Neural Networks: time $= t_1$.



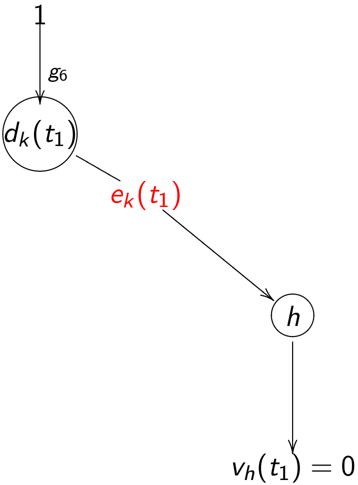
$w_{ik}(t_1) = v_i(t_1) = g_6$ is the Gödel number of $P(x)$;
$d_k(t_1) = g_1$ is the Gödel number of $P(0)$.

Example of Unification in Neural Networks: time $= t_1$.



$w_{ik}(t_1) = v_i(t_1) = g_6$ is the Gödel number of $P(x)$;
$d_k(t_1) = g_1$ is the Gödel number of $P(0)$.

Example of Unification in Neural Networks: time $= t_1$.

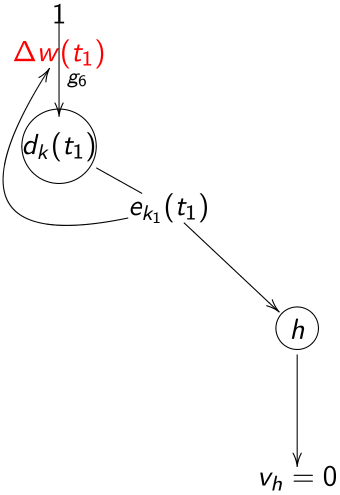


$w_{ki}(t_1) = v_i(t_1) = g_6$ is the Gödel number of $P(x)$;

$d_k(t_1) = g_1$ is the Gödel number of $P(0)$;

Compute $e_k(t_1) = s(d_k(t_1) \ominus v_k(t_1))$ - the Gödel number of substitution for the disagreement set $d_k(t_1) \ominus v_k(t_1)$.

Example of Unification in Neural Networks: time $= t_1$.



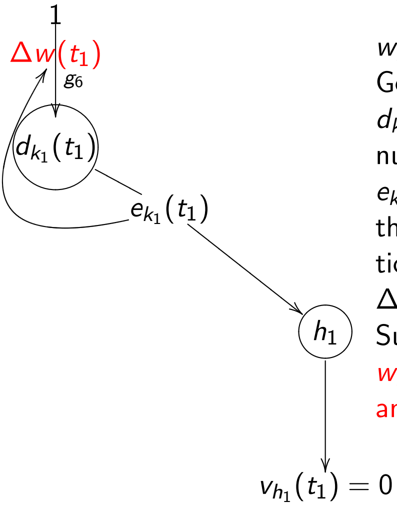
$w_{ki}(t_1) = v_k(t_1) = g_6$ is the Gödel number of $P(x)$;

$d_k(t_1) = g_1$ is the Gödel number of $P(0)$;

$e_k(t_1) = s(d_k(t_1) \ominus v_k(t_1))$ - the Gödel number of substitution for the disagreement set $d_k(t_1) \ominus v_k(t_1)$;

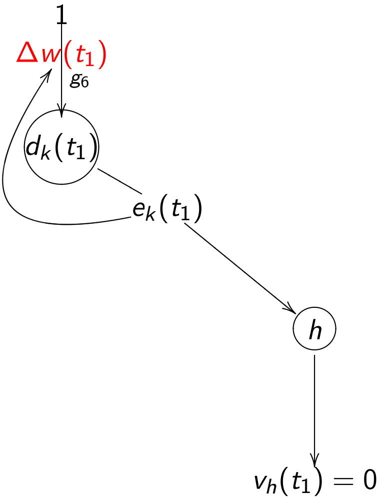
$\Delta w(t_1) = v_i(t_1)e_k(t_1) = e_k(t_1)$.

Example of Unification in Neural Networks: time $= t_1$.



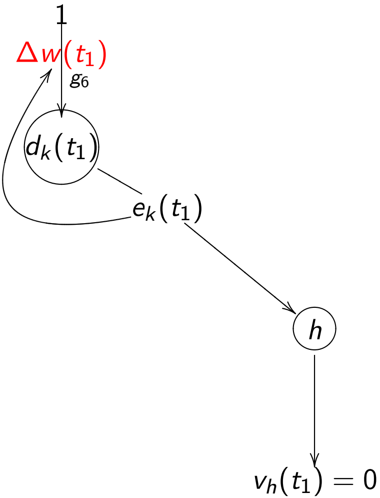
$w_{ki}(t_1) = v_k(t_1) = g_6$ is the Gödel number of $P(x)$;
$d_k(t_1) = g_1$ is the Gödel number of $P(0)$;
$e_k(t_1) = s(d_k(t_1) \ominus v_k(t_1))$ - the Gödel number of substitution x_1/a_1;
$\Delta w(t_1) = v_i(t_1)e_k(t_1)$;
Substitutions are applied: $w_{ki}(t_2) = w_{ki}(t_1) \odot \Delta w(t_1)$ and $d_k(t_2) = d_k(t_1) \odot \Delta w(t_1)$.

Example of Unification in Neural Networks: time $= t_{1-2}$.



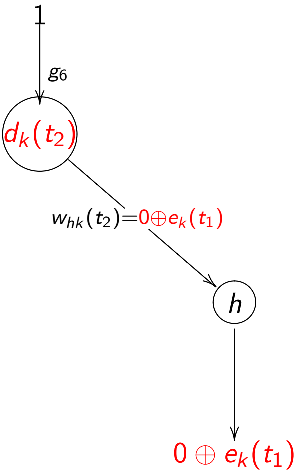
$w_{ki}(t_1) = v_k(t_1) = g_6$ is the Gödel number of $P(x)$;
$d_k(t_1) = g_1$ is the Gödel number of $P(0)$;
$e_k(t_1) = s(d_k(t_1) \ominus v_k(t_1))$ - the Gödel number of substitution x_1/a_1;
$\Delta w(t_1) = v_i(t_1) e_k(t_1)$;
The substitutions are applied:
$w_{ki}(t_2) = w_{ki}(t_1) \odot \Delta w(t_1)$ and $d_k(t_2) = d_k(t_1) \odot \Delta w(t_1)$.

Example of Unification in Neural Networks: time $= t_{1-2}$.



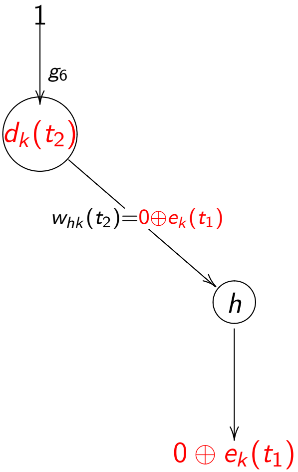
$w_{ki}(t_1) = v_k(t_1) = g_6$ is the Gödel number of $P(x)$;
$d_k(t_1) = g_1$ is the Gödel number of $P(0)$;
$e_k(t_1) = s(d_k(t_1) \ominus v_k(t_1))$ - the Gödel number of substitution x_1/a_1;
$\Delta w(t_1) = v_i(t_1)e_k(t_1)$;
The substitutions are applied:
$w_{ki}(t_2) = w_{ki}(t_1) \odot \Delta w(t_1)$ and $d_k(t_2) = d_k(t_1) \odot \Delta w(t_1)$.
$w_{hk}(t_2) = w_{hk}(t_1) \oplus e_k(t_1)$.

Example of Unification in Neural Networks: time $= t_2$.



$w_{ik}(t_2) = v_i(t_2) = g_6$ is the Gödel number of $P(0)$;
$d_k(t_2) = g_7$ is the Gödel number of $P(0)$.

Example of Unification in Neural Networks: time $= t_2$.

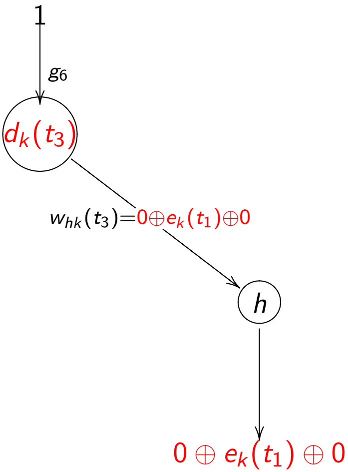


$w_{ik}(t_2) = v_i(t_2) = g_6$ is the Gödel number of $P(0)$;

$d_k(t_2) = g_7$ is the Gödel number of $P(0)$.

$v_i(t_2) \ominus d_k(t_2) = \emptyset$. This means that we set $e_k(t_2) = 0$.

Example of Unification in Neural Networks: time $= t_3$.



$w_{hk}(t_3) = w_{hk}(t_2) \oplus 0;$
$v_h(t_3) = w_{hk}(t_3).$
When v_h starts and ends with 0, computation stops.

Preliminary conclusions

Properties of these neural networks

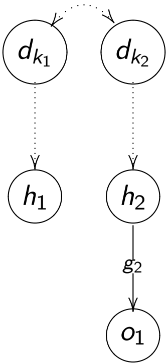
- First-order atoms are embedded directly into a neural network via Gödel numbers.
- Neural networks are finite and give deterministic results, comparing with infinite layers needed to perform substitutions in [HK94].
- Unification algorithm is performed as an adaptive process, which corrects one piece of data relatively to the other piece of data.

Main theorem

Theorem

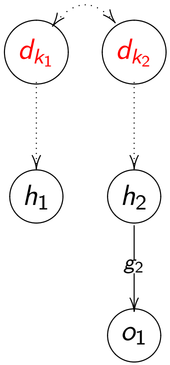
Let P be a definite logic program and G be a definite goal. Then there exists a 3-layer recurrent neural network which computes the Gödel number s of substitution θ if and only if SLD-refutation derives θ as an answer for $P \cup \{G\}$. (We will call these neural networks SLD neural networks).

Example. Time = t_1.



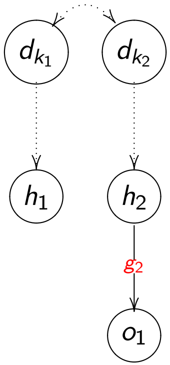
$$P(0) \leftarrow;$$
$$P(s(x)) \leftarrow P(x).$$

Example. Time = t_1.



$P(0) \leftarrow;$
$P(s(x)) \leftarrow P(x).$

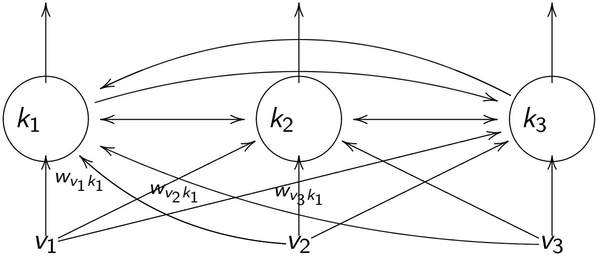
Example. Time $= t_1$.



$$P(0) \leftarrow;$$
$$P(s(x)) \leftarrow P(x).$$

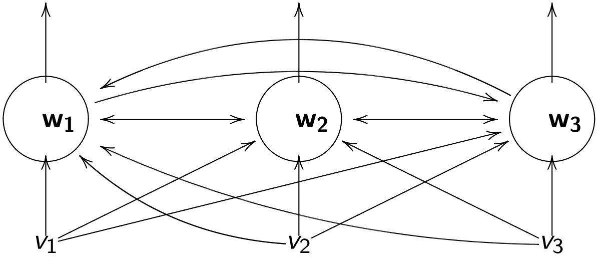
Competitive learning; Kohonen's layer

We compute additional parameter $I_i = D(\mathbf{w_i}, \mathbf{v})$,
$D(\mathbf{w_i}, \mathbf{v})$ is the distance measurement function.
The common choice for $D(\mathbf{w_i}, \mathbf{v})$ is the Euclidian distance $|w_i - v|$.



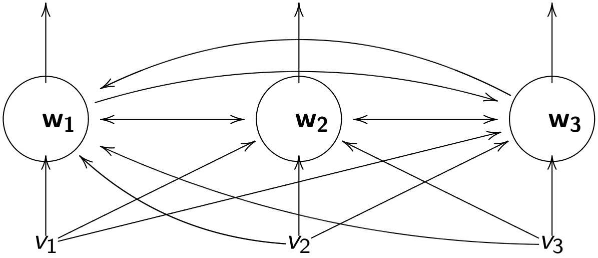
Competitive learning; Kohonen's layer

We compute additional parameter $I_i = D(\mathbf{w_i}, \mathbf{v})$,
$D(\mathbf{w_i}, \mathbf{v})$ is the distance measurement function.
The common choice for $D(\mathbf{w_i}, \mathbf{v})$ is the Euclidian distance $|w_i - v|$.



Competitive learning; Kohonen's layer

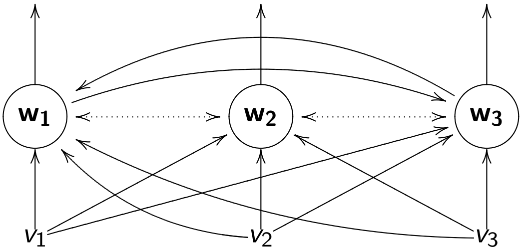
We compute additional parameter $I_i = D(\mathbf{w_i}, \mathbf{v})$,
$D(\mathbf{w_i}, \mathbf{v})$ is the distance measurement function.
The common choice for $D(\mathbf{w_i}, \mathbf{v})$ is the Euclidian distance $|w_i - v|$.



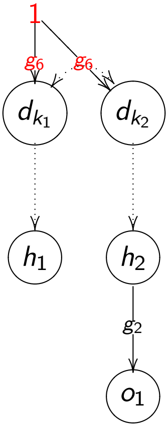
We use a reduced form of this learning rule. With $I_i = i$, we use it to encode the order of clauses; and hence the priority among neurons.

Competitive learning; Kohonen's layer

We will denote the Kohonen's layer by $\leftarrow\cdots\rightarrow$:

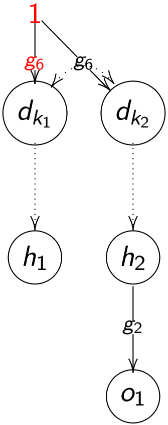


Example. Time = t_1.



$G_0 = g_6 = P(x)$.
$P(0) \leftarrow$;
$P(s(x)) \leftarrow P(x)$.

Example. Time $= t_1$.



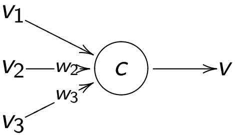
$G_0 = g_6 = P(x)$.
$P(0) \leftarrow$;
$P(s(x)) \leftarrow P(x)$.

Filter Learning: Grossberg's law

Grossberg's law is expressed by the equation

$$w_{ci}^{\text{new}} = w_{ci}^{\text{old}} + [v_i v_1 - w_{ci}^{\text{old}}]U(v_i), (i \in \{2, 3\}),$$

where $U(v_i) = 1$ if $v_i > 0$ and $U(v_i) = 0$ otherwise.

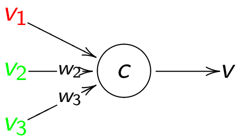


Filter Learning: Grossberg's law

Grossberg's law is expressed by the equation

$$w_{ci}^{\text{new}} = w_{ci}^{\text{old}} + [v_i v_1 - w_{ci}^{\text{old}}]U(v_i), (i \in \{2, 3\}),$$

where $U(v_i) = 1$ if $v_i > 0$ and $U(v_i) = 0$ otherwise.

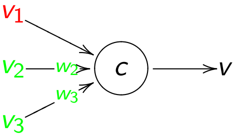


Filter Learning: Grossberg's law

Grossberg's law is expressed by the equation

$$w_{ci}^{\text{new}} = w_{ci}^{\text{old}} + [v_i v_1 - w_{ci}^{\text{old}}]U(v_i), (i \in \{2, 3\}),$$

where $U(v_i) = 1$ if $v_i > 0$ and $U(v_i) = 0$ otherwise.

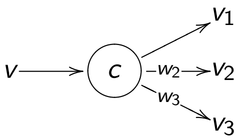


Filter Learning: inverse Grossberg's law

The inverse form of Ginsberg's law:

$$w_{ic}^{\mathrm{new}} = w_{ic}(t)^{\mathrm{old}} + [v_i \, v_1 - w_{ic}^{\mathrm{old}}]U(v_i), (i \in \{2, 3\}),$$

where $U(v_i) = 1$ if $v_i > 0$ and $U(v_i) = 0$ otherwise.

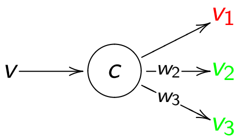


Filter Learning: inverse Grossberg's law

The inverse form of Ginsberg's law:

$$w_{ic}^{\mathrm{new}} = w_{ic}(t)^{\mathrm{old}} + [v_i \, v_1 - w_{ic}^{\mathrm{old}}] U(v_i), (i \in \{2, 3\}),$$

where $U(v_i) = 1$ if $v_i > 0$ and $U(v_i) = 0$ otherwise.

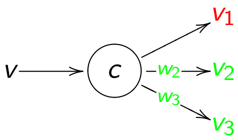


Filter Learning: inverse Grossberg's law

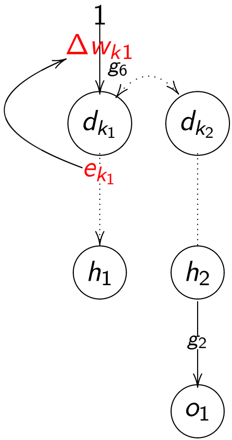
The inverse form of Ginsberg's law:

$$w_{ic}^{\mathrm{new}} = w_{ic}(t)^{\mathrm{old}} + [v_i v_1 - w_{ic}^{\mathrm{old}}]U(v_i), (i \in \{2, 3\}),$$

where $U(v_i) = 1$ if $v_i > 0$ and $U(v_i) = 0$ otherwise.

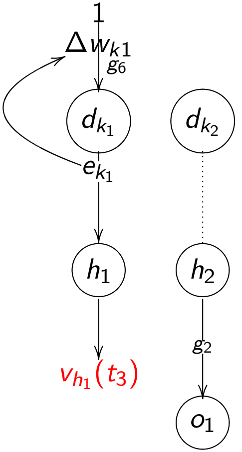


Example. Time t_1: signals are filtered and unification initialized.



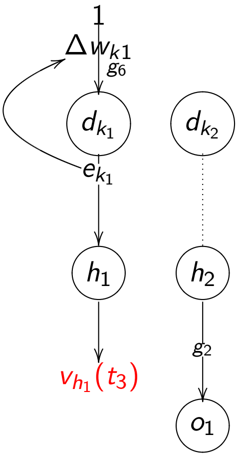
$G_0 = g_6 = P(x).$
$P(0) \leftarrow;$
$P(s(x)) \leftarrow P(x).$

Example. Time $t_2 - t_3$: unification.



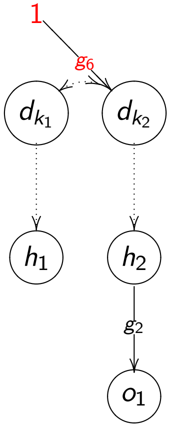
$g_6 = P(x)$.
$P(0) \leftarrow$;
$P(s(x)) \leftarrow P(x)$.

Example. Time $t_2 - t_3$: unification.



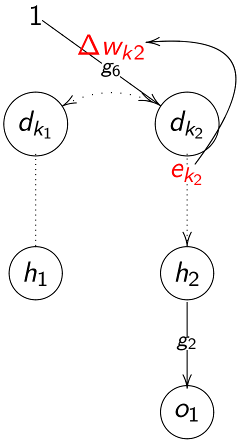
$g_6 = P(x)$.
$P(0) \leftarrow$;
$P(s(x)) \leftarrow P(x)$.
$v_{h_1}(t_3) = 0 \oplus e_{k_1}(t_1) \oplus 0$.
That is, the output is the Gödel number for the substitution $x/0$.
Computations terminate.

Example. Time $= t_4$.



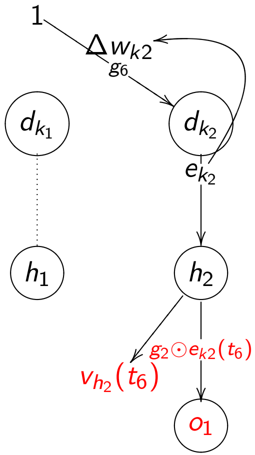
$G_0 = g_6 = P(x)$.
$P(0) \leftarrow$;
$P(s(x)) \leftarrow P(x)$.

Example. Time $t_4 - t_6$: unification.



$G_0 = g_6 = P(x)$.
$P(0) \leftarrow$;
$P(s(x)) \leftarrow P(x)$.
Using the error-correction learning, the network computes $e_{k2}(t_6)$, the Gödel number of the substitution $(x/s(x))$.

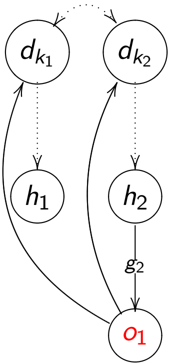
Example. Time t_6: next step.



$P(0) \leftarrow;$
$P(s(x)) \leftarrow P(x).$

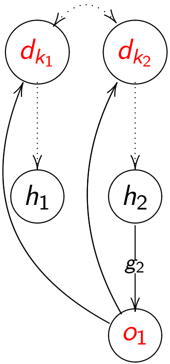
The signal $e_{k2}(t_6)$ is given as an output value $v_{h_2}(t_6)$; it is also used to amend and activate the weight g_2; the signal passes to the neuron o_1.

Example. Time $= t_7$. New iteration starts.



$P(0) \leftarrow;$
$P(s(x)) \leftarrow P(x).$
The signal $g_2 \odot e_{k2}(t_6)$
is sent from the unit o_1
to the input units.
It is the Gödel number of
$P(x)\theta = P(s(x))$.
the signals will be filtered;
and only one of them
will be processed next.

Example. Time = t_7. New iteration starts.



$P(0) \leftarrow;$
$P(s(x)) \leftarrow P(x).$
The signal $g_2 \odot e_{k2}(t_6)$
is sent from the unit o_1
to the input units.
It is the Gödel number of
$P(x)\theta = P(s(x)).$
the signals will be filtered;
and only one of them
will be processed next.

Conclusions

- SLD neural networks have finite architecture, but their effectiveness is due to several learning functions.
- Unification is performed as adaptive process.
- Atoms and substitutions are represented in SLD neural networks directly, via Gödel numbers, and hence allow easier machine implementations.

Future Work

- Practical implementations of SLD neural networks.

Future Work

- Practical implementations of SLD neural networks.
- Theoretical development:
    - SLD neural networks allow higher-order generalizations.
    - ...can therefore be extended to higher-order Horn logics, hereditary Harrop logics...
    - ...can be extended to non-classical logic programs: linear, many-valued, etc...
    - Inductive logic and SLD neural networks.

Thank you!