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

Outline: The Core Method for First Order Logic

The Core Method

- Relate logic programs and connectionist systems

$$I_{\mathcal{L}} \xrightarrow{T_P} I_{\mathcal{L}}$$

The Core Method

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

$$\begin{tikzcd}
I_{\mathcal{L}} \arrow[r, "T_P"] \arrow[d, "\iota"] & I_{\mathcal{L}} \\
\mathbb{R}^m & \mathbb{R}^m \arrow[u, "\iota^{-1}"']
\end{tikzcd}$$

The Core Method

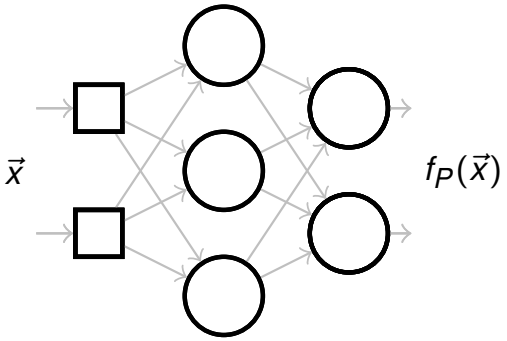
- Relate logic programs and connectionist systems
- Embed interpretations into (vectors of) real numbers.
- Hence, obtain an embedded version of the T_P-operator.

$$\begin{tikzcd}
I_{\mathcal{L}} \arrow[r, "T_P"] \arrow[d, "\iota"] & I_{\mathcal{L}} \\
\mathbb{R}^m \arrow[r, "f_P"] & \mathbb{R}^m \arrow[u, "\iota^{-1}"']
\end{tikzcd}$$

The Core Method

- Relate logic programs and connectionist systems
- Embed interpretations into (vectors of) real numbers.
- Hence, obtain an embedded version of the T_P-operator.
- Construct a network computing one application of f_P.

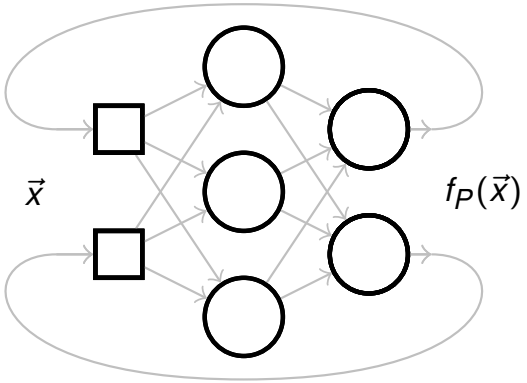
$$\begin{tikzcd}
I_{\mathcal{L}} \arrow[r, "T_P"] \arrow[d, "\iota"] & I_{\mathcal{L}} \\
\mathbb{R}^m \arrow[r, "f_P"] & \mathbb{R}^m \arrow[u, "\iota^{-1}"']
\end{tikzcd}$$



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.

$$\begin{tikzcd}
I_{\mathcal{L}} \arrow[r, "T_P"] \arrow[d, "\iota"] & I_{\mathcal{L}} \\
\mathbb{R}^m \arrow[r, "f_P"] & \mathbb{R}^m \arrow[u, "\iota^{-1}"']
\end{tikzcd}$$



First Order Logic Programs – Two Examples

$nat(0).$ *% 0 is a natural number.*

$nat(succ(X)) \leftarrow nat(X).$ *% The successor $succ(X)$ is a natural*
% number if X is a natural number.

$even(0).$ *% 0 is an even number.*

$even(succ(X)) \leftarrow odd(X).$ *% The successor of an odd X is even.*

$odd(X) \leftarrow \neg even(X).$ *% If X is not even then it is odd.*

First Order Logic Programs – The Syntax

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

Functions, Variables and Terms

$$\mathcal{F} = \{0/0, succ/1\}$$
$$\mathcal{V} = \{X\}$$
$$\mathcal{T} = \{0, succ(0), succ(X), succ(succ(0)), \ldots\}$$

Predicate Symbols and Atoms

$$\mathcal{P} = \{even/1, odd/1\}$$
$$\mathcal{A} = \{even(succ(X)), odd(succ(0)), odd(0), odd(X), \ldots\}$$

Connectives, Clause and Program

$\nearrow$ propositional logic

First Order Logic Programs – The Semantics

Herbrand Base $B_{\mathcal{L}}$ = Set of ground atoms

$$B_{\mathcal{L}} = \{even(0), even(succ(0)), \ldots, odd(0), odd(succ(0)), \ldots\}$$

Interpretations = Subsets of the Herbrand base

$$I_1 = \{even(succ^{2n}(0) \mid n \geq 1\} \qquad I_2 = \{\}$$

$$I_3 = \{odd(succ^{2n+1}(0) \mid n \geq 0\} \qquad I_4 = I_1 \cup I_3$$

$I_{\mathcal{L}}$ = Space of all Interpretations

T_P for our running examples

Definition (T_P)

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

Example (Natural numbers)

$n(0).$
$n(s(X)) \leftarrow n(X).$

$$\begin{aligned} \{\} &\mapsto \{n(0)\} \\ \{n(0)\} &\mapsto \{n(0), n(s(0))\} \\ \{n(0), n(s(0))\} &\mapsto \{n(0), n(s(0)), n(s(s(0)))\} \\ \{n(X) \mid X \in \mathcal{T}\} &\mapsto \{n(X) \mid X \in \mathcal{T}\} \end{aligned}$$

T_P for our running examples

Definition (T_P)

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

Example (Natural numbers)

$n(0).$
$n(s(X)) \leftarrow n(X).$

$$\begin{aligned}
\{\} &\mapsto \{n(0)\} \\
\{n(0)\} &\mapsto \{n(0), n(s(0))\} \\
\{n(0), n(s(0))\} &\mapsto \{n(0), n(s(0)), n(s(s(0)))\} \\
\{n(X) \mid X \in \mathcal{T}\} &\mapsto \{n(X) \mid X \in \mathcal{T}\}
\end{aligned}$$

Example (Even and odd numbers)

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

$$\begin{aligned}
\{\} &\mapsto \{e(0), o(X) \mid X \in \mathcal{T}\} \\
\{o(X) \mid X \in \mathcal{T}\} &\mapsto \{e(0), e(s(X)), o(X) \mid X \in \mathcal{T}\} \\
\{e(s^{2n}(0)) \mid n \geq 0\} &\mapsto \{e(0), o(s^{2n+1}(0)) \mid n \geq 0\} \\
\{o(s^{2n+1}(0)) \mid n \geq 0\} &\mapsto \{e(0), e(s^{2n}(0)) \mid n \geq 0\} \\
B_{\mathcal{L}} &\mapsto \{e(0), e(s(X)) \mid X \in \mathcal{T}\}
\end{aligned}$$

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

Level Mappings

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

Example (Even and odd numbers)

$$|e(s^n(0))| = 2n + 1 \qquad |o(s^n(0))| = 2n + 2$$

Level Mappings

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

  Example (Even and odd numbers)

$$|e(s^n(0))| = 2n + 1 \qquad |o(s^n(0))| = 2n + 2$$

- ... hence, enumerates the Herbrand base:

  Example (Even and odd numbers)

$$[\underbrace{e(0)}_{1}, \underbrace{o(0)}_{2}, \underbrace{e(s(0))}_{3}, \underbrace{o(s(0))}_{4}, \underbrace{e(s(s(0)))}_{5}, \ldots]$$

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:

$$B_{\mathcal{L}} = [\ e(0), o(0), e(1), o(1), e(2), \ldots]$$

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:

$$\begin{array}{rl} B_{\mathcal{L}} = & [\ e(0), o(0), e(1), o(1), e(2), \ldots] \\ \hline \iota(\{e(0)\}) = & 0.\ 1\ \ 0\ \ 0\ \ 0\ \ 0_4 \end{array}$$

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:

$$\begin{array}{rl} B_{\mathcal{L}} = & [\ e(0), o(0), e(1), o(1), e(2), \ldots] \\ \hline \iota(\{e(0)\}) = & 0.\ 1\ \ 0\ \ 0\ \ 0\ \ 0_4 = 0.25_{10} \end{array}$$

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:

	$B_{\mathcal{L}} =$ [	$e(0)$,	$o(0)$,	$e(1)$,	$o(1)$,	$e(2)$,	$\ldots$]
$\iota(\{e(0)\}) =$	$0.$	1	0	0	0	0_4	$= 0.25_{10}$
$\iota(\{e(0), e(1), e(2)\}) =$	$0.$	1	0	1	0	1_4	$\approx 0.27_{10}$

$\mathfrak{C}$ - The Set of all embedded Interpretations

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

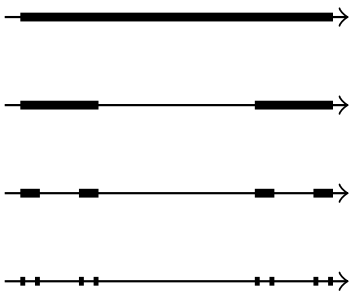
$$\mathfrak{C} = \{\iota(I) \mid I \in I_{\mathcal{L}}\}$$

$\mathfrak{C}$ - The Set of all embedded Interpretations

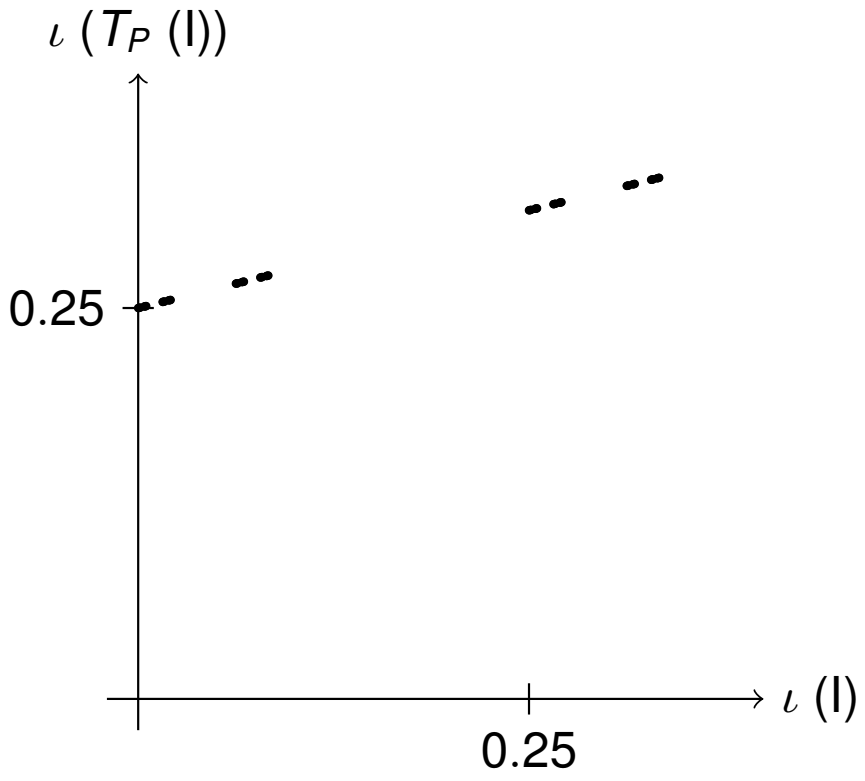
- $\mathfrak{C}$ for the 1-dimensional case:

$$\mathfrak{C} = \{\iota(I) \mid I \in I_{\mathcal{L}}\}$$

- Another construction:



The Graph of the Natural Numbers

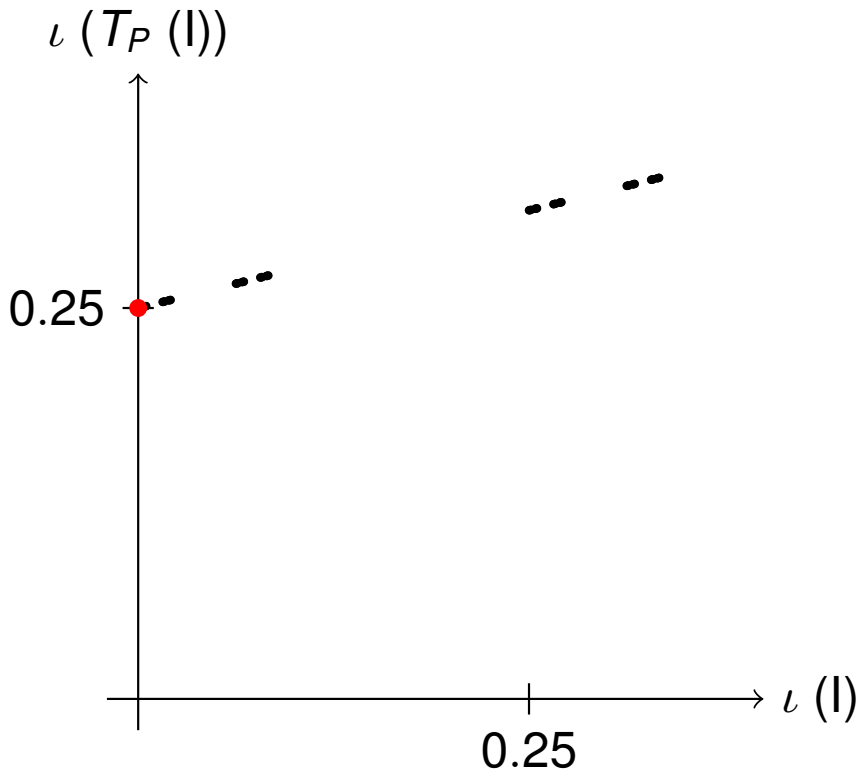


$n(0).$

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

$|n(s^n(0))| = n + 1$

The Graph of the Natural Numbers



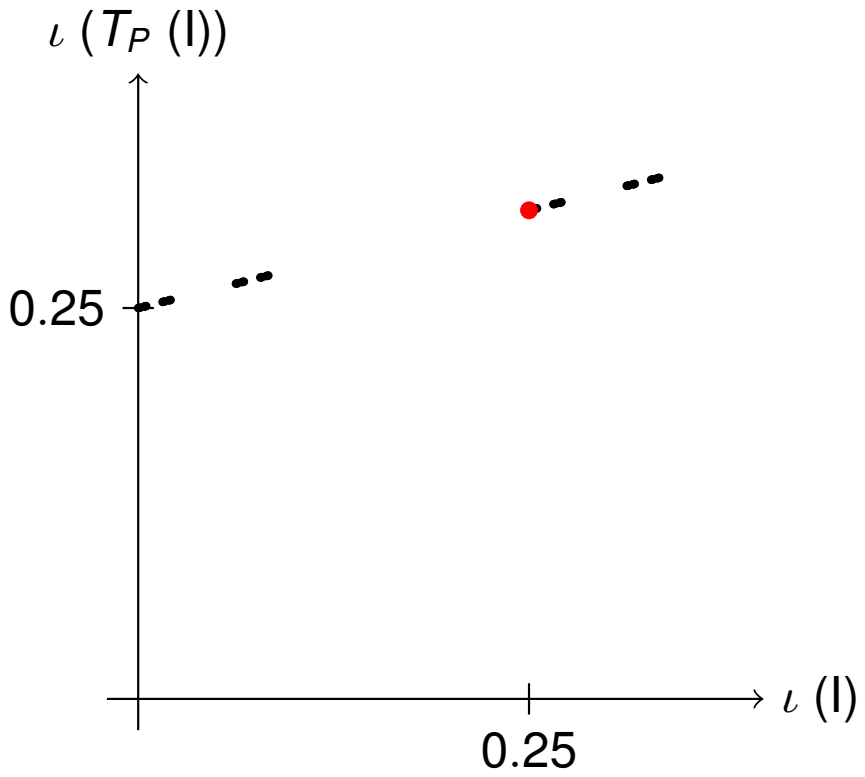
$n(0).$

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

$|n(s^n(0))| = n + 1$

$$\underbrace{\{\}}_{0.0} \mapsto \underbrace{\{n(0)\}}_{0.25}$$

The Graph of the Natural Numbers



$n(0).$

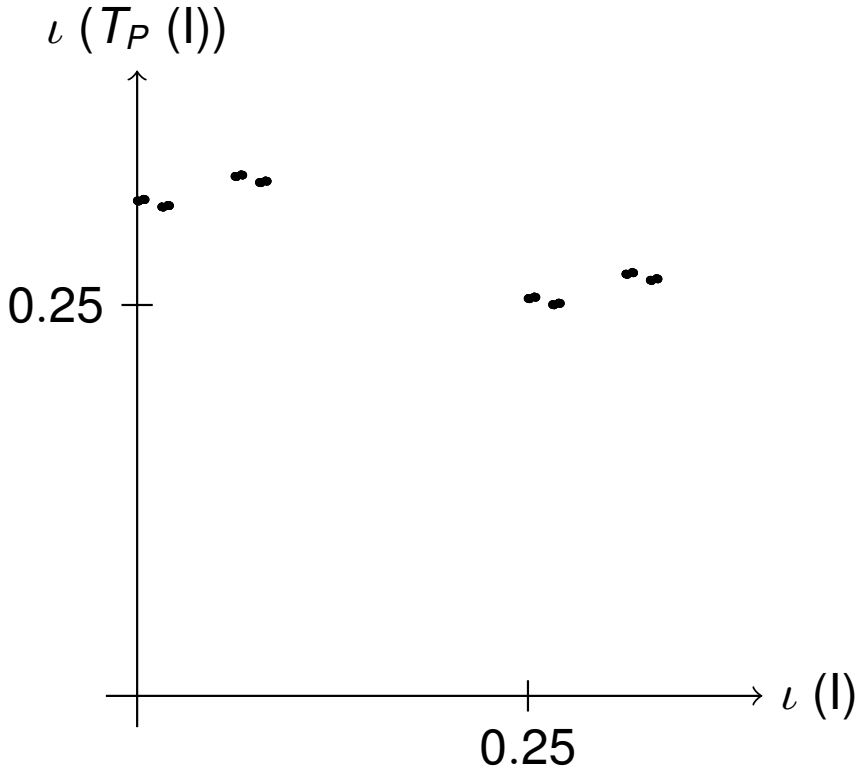
$n(s(X)) \leftarrow n(X).$

$$|n(s^n(0))| = n + 1$$

$$\underbrace{\{\}}_{0.0} \mapsto \underbrace{\{n(0)\}}_{0.25}$$

$$\underbrace{\{n(0)\}}_{0.25} \mapsto \underbrace{\{n(0), n(s(0))\}}_{0.3125}$$

The Graph of the Even and Odd Numbers



$e(0).$

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

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

Continuity of the Embedded T_P-Operator

- The mapping ι is ...
  - is continuous
  - is a bijection between $I_{\mathcal{L}}$ and $\mathfrak{C}$
  - has a continuous inverse ι^{-1}
  - hence, is a homeomorphism
- The set $I_{\mathcal{L}}$ together with a suitable metric is compact $\rightsquigarrow$ $\mathfrak{C}$ is compact.
- T_P is continuous for certain programs $\rightsquigarrow$ f_P is continuous
- We can apply Funahashis theorem.

Some Results

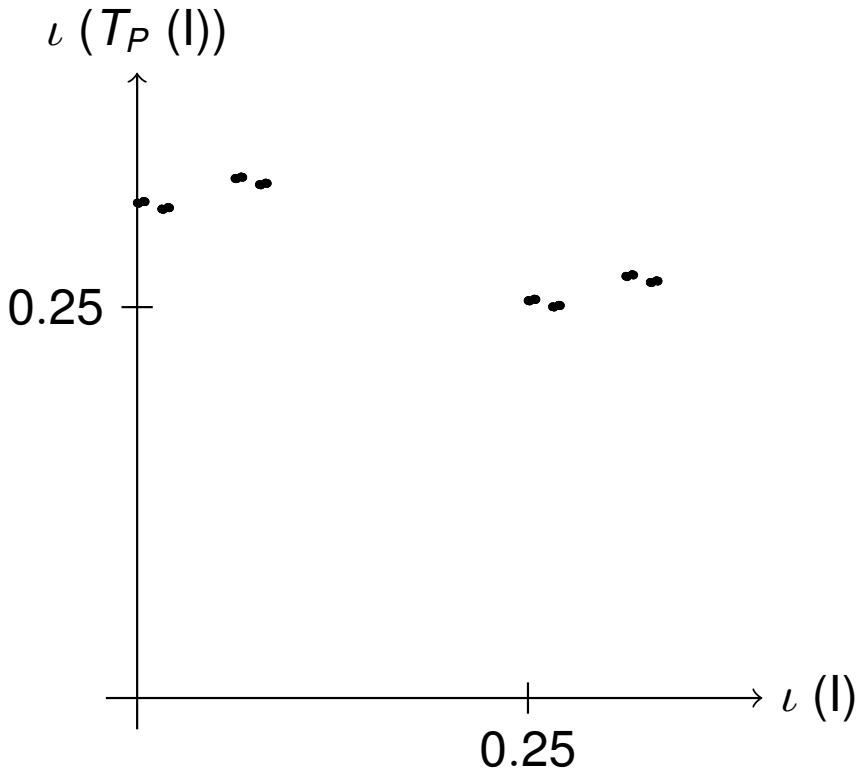
Theorem (Hölldobler, Kalinke & Störr, 1999)

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 limitted class of logic programs.

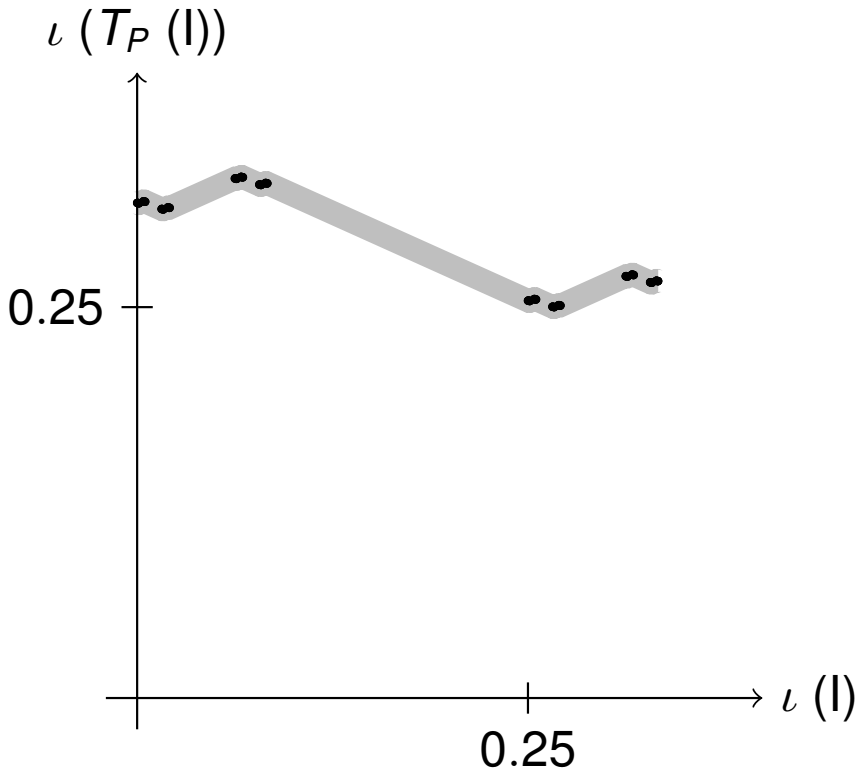
Approximating the Embedded T_P-Operator



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

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

Approximating the Embedded T_P-Operator

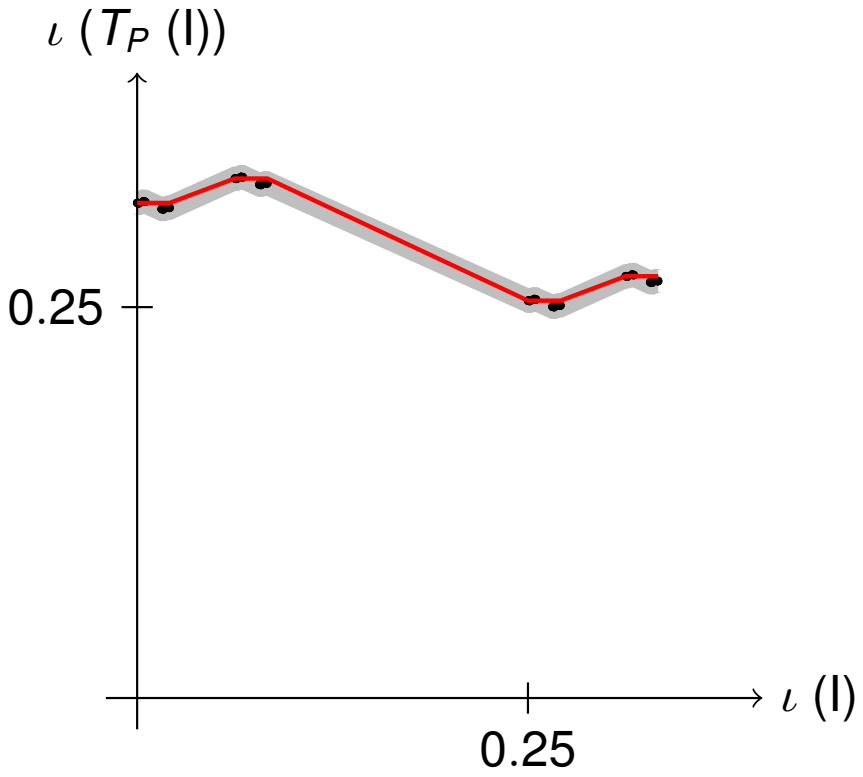


$e(0).$
$e(s(X)) \leftarrow o(X).$
$o(X) \leftarrow \neg e(X).$

$\varepsilon = 0.05$

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

Approximating the Embedded T_P-Operator



$e(0).$

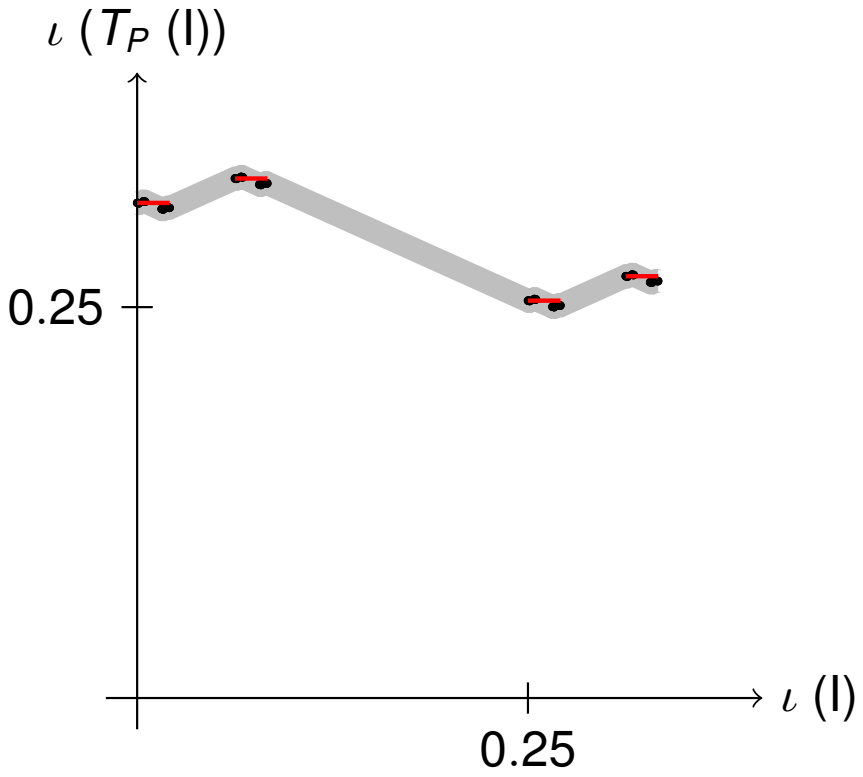
$e(s(X)) \leftarrow o(X).$

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

$\varepsilon = 0.05$

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

Approximating the Embedded T_P-Operator



$e(0).$

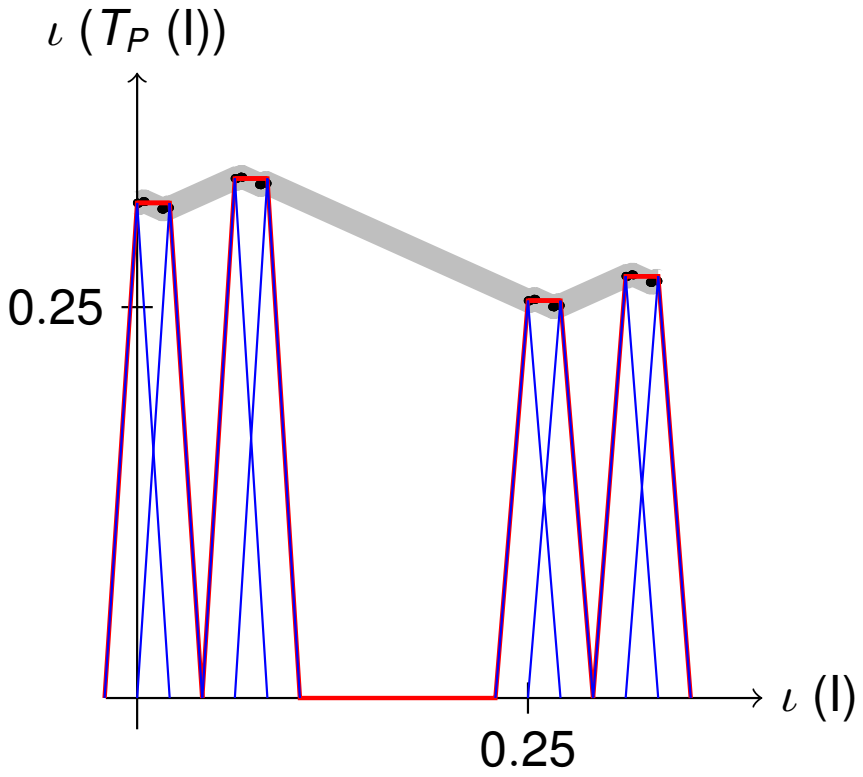
$e(s(X)) \leftarrow o(X).$

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

$\varepsilon = 0.05$

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

Approximating the Embedded T_P-Operator



$e(0).$

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

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

$\varepsilon = 0.05$

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

A Problem ...

- The accuracy of this approach is very limitted.
- 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

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)

$$\|e(s^n(0))\| = (n+1, 1) \qquad \|o(s^n(0))\| = (n+1, 2)$$

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)

$$\|e(s^n(0))\| = (n+1, 1) \qquad \|o(s^n(0))\| = (n+1, 2)$$

- ... still "enumerates" the Herbrand base:

  Example (Even and odd numbers)

	1	2	3	4
$dim1:$	$e(0)$	$e(s(0))$	$e(s(s(0)))$	$e(s(s(s(0))))$
$dim2:$	$o(0)$	$o(s(0))$	$o(s(s(0)))$	$o(s(s(s(0))))$

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:

$$\vec{\iota}(I) = \sum_{A \in I} \vec{\iota}(A)$$

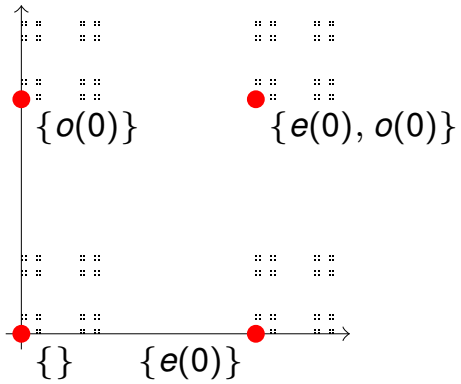
$$\vec{\iota}(A) = (\iota_1(A), \ldots, \iota_m(A)) \text{ with}$$

$$\iota_i(A) = \begin{cases} 4^{-l} & \text{for } \|A\| = (l, d) \text{ and } i = d \\ 0 & \text{otherwise} \end{cases}$$

$\mathfrak{C}$ - The Set of all embedded Interpretations

- $\mathfrak{C}$ for the 2-dimensional case:

$$\mathfrak{C} = \{\vec{\iota}(I) \mid I \in I_{\mathcal{L}}\}$$

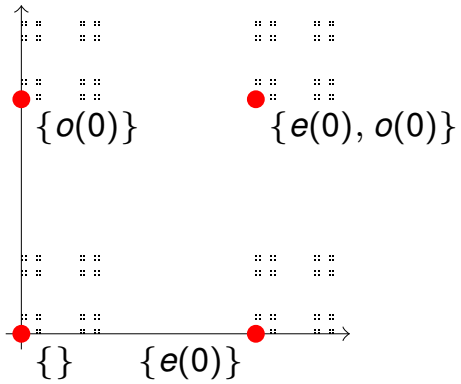


- Another construction:

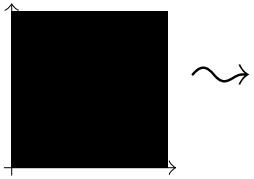
$\mathfrak{C}$ - The Set of all embedded Interpretations

- $\mathfrak{C}$ for the 2-dimensional case:

$$\mathfrak{C} = \{\vec{\iota}(I) \mid I \in I_{\mathcal{L}}\}$$



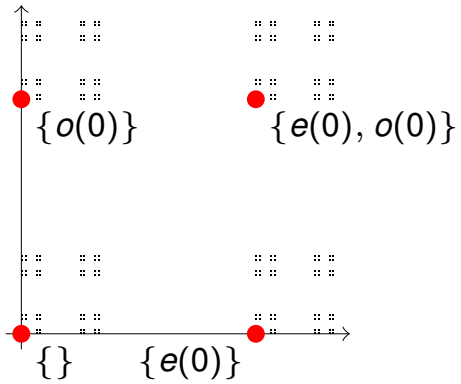
- Another construction:



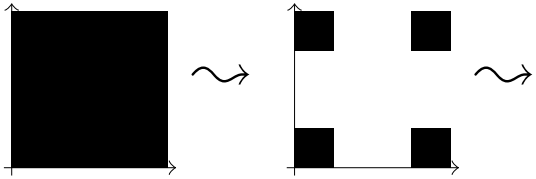
$\mathfrak{C}$ - The Set of all embedded Interpretations

- $\mathfrak{C}$ for the 2-dimensional case:

$$\mathfrak{C} = \{\vec{\iota}(I) \mid I \in I_{\mathcal{L}}\}$$



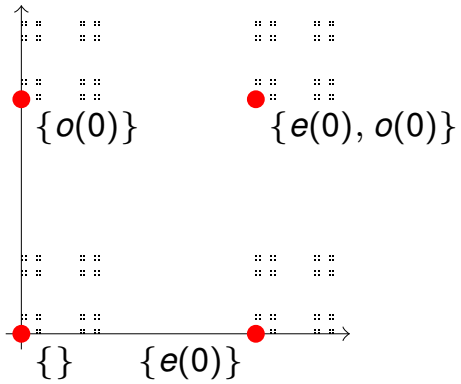
- Another construction:



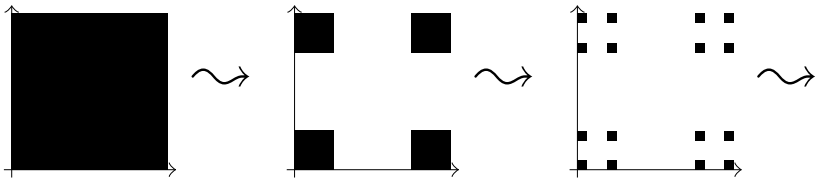
$\mathfrak{C}$ - The Set of all embedded Interpretations

- $\mathfrak{C}$ for the 2-dimensional case:

$$\mathfrak{C} = \{\vec{\iota}(I) \mid I \in I_{\mathcal{L}}\}$$



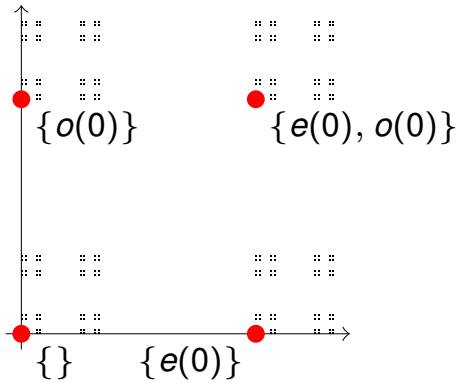
- Another construction:



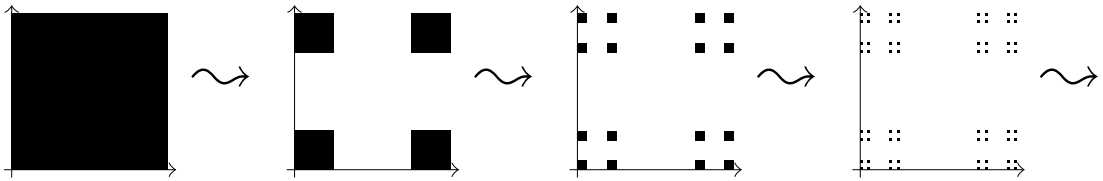
$\mathfrak{C}$ - The Set of all embedded Interpretations

- $\mathfrak{C}$ for the 2-dimensional case:

$$\mathfrak{C} = \{\vec{\iota}(I) \mid I \in I_{\mathcal{L}}\}$$



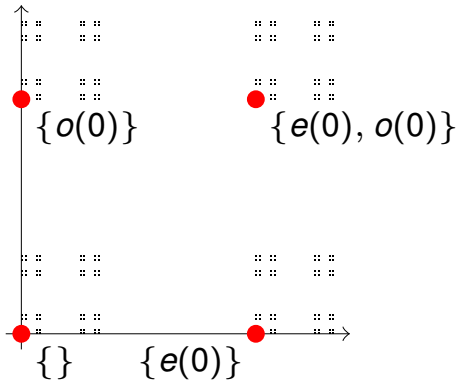
- Another construction:



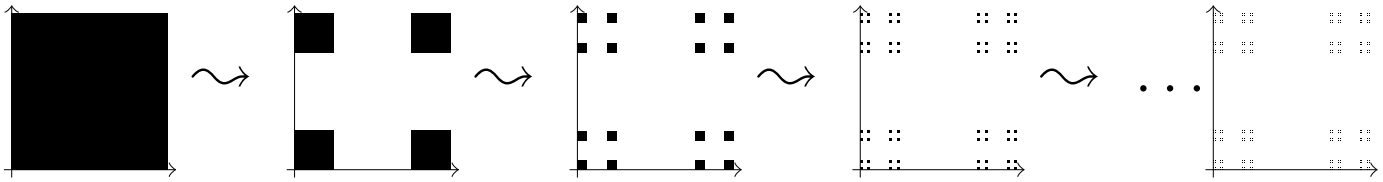
$\mathfrak{C}$ - The Set of all embedded Interpretations

- $\mathfrak{C}$ for the 2-dimensional case:

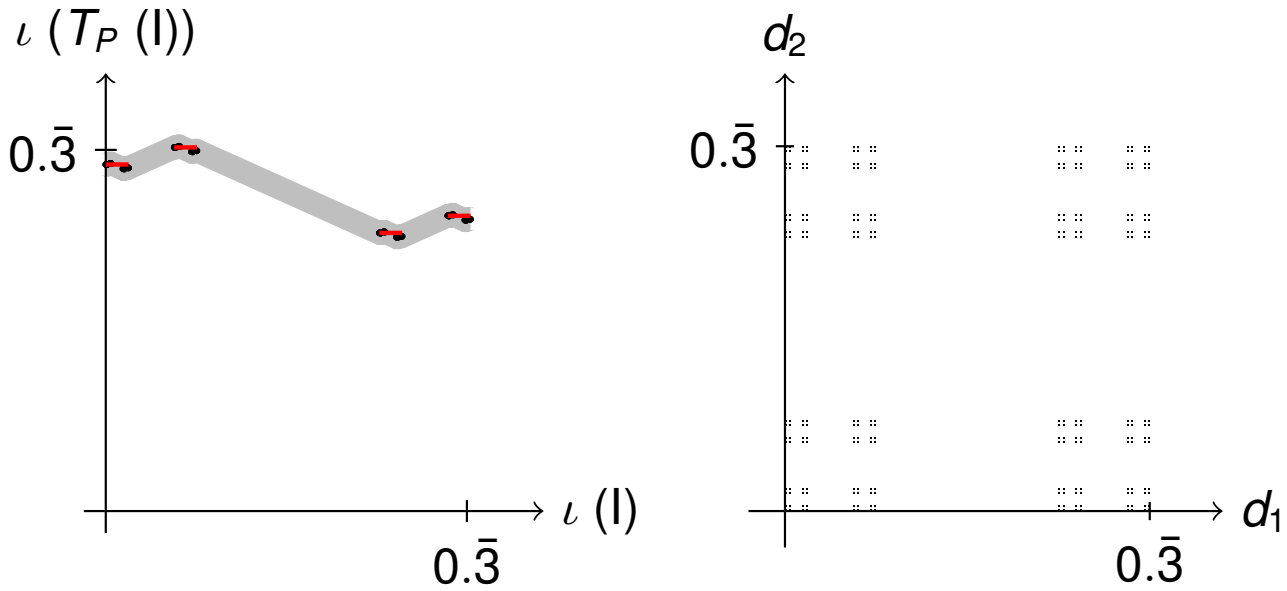
$$\mathfrak{C} = \{\vec{\iota}(I) \mid I \in I_{\mathcal{L}}\}$$



- Another construction:



Approximating the Embedded T_P-Operator



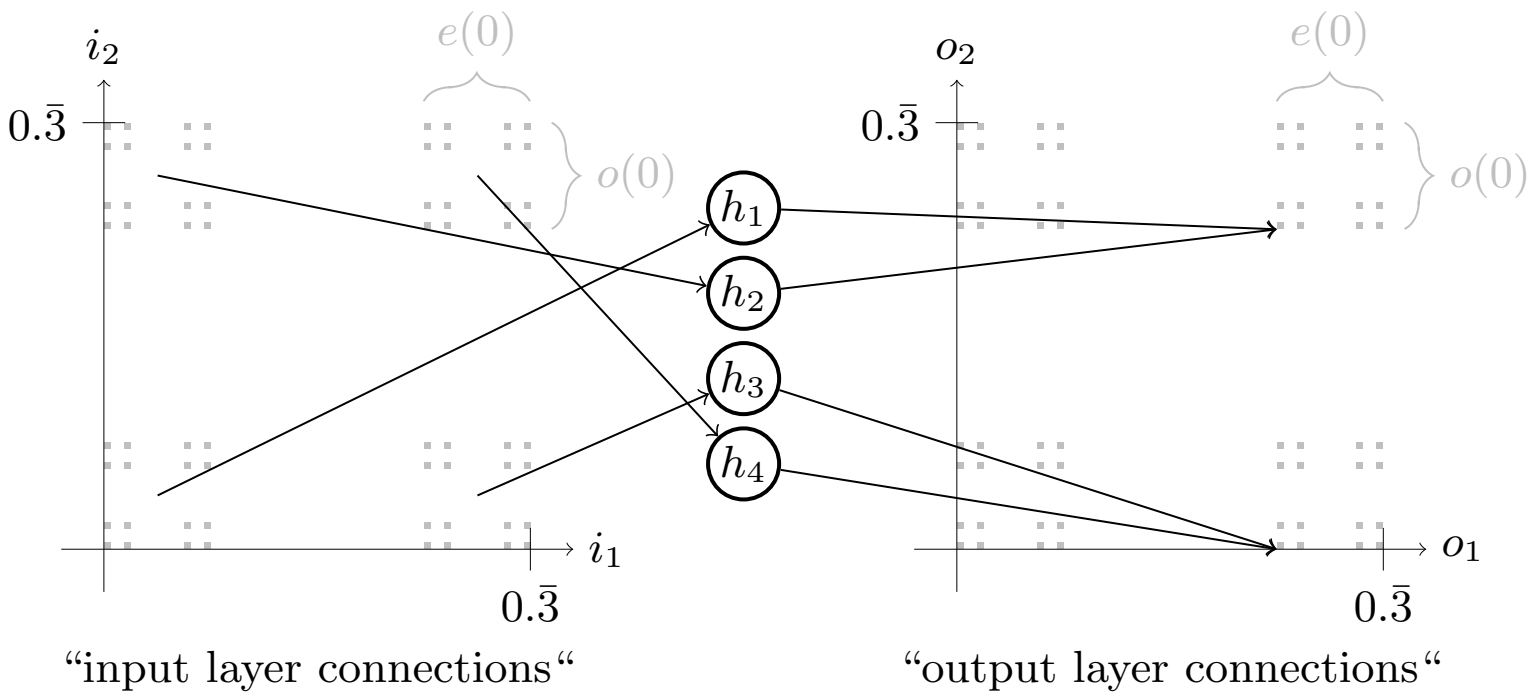
Implementation

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.

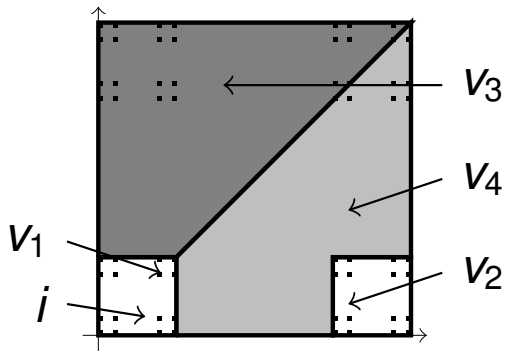
The FineBlend

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



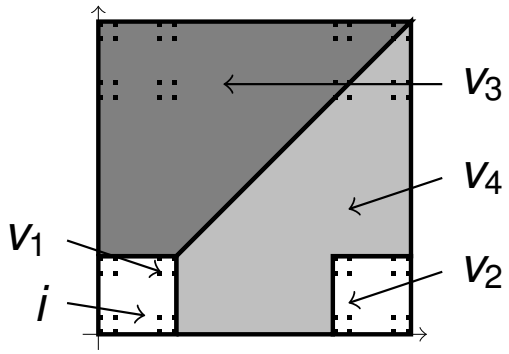
FineBlend: Training

- Incoming weights encode *responsible areas*:

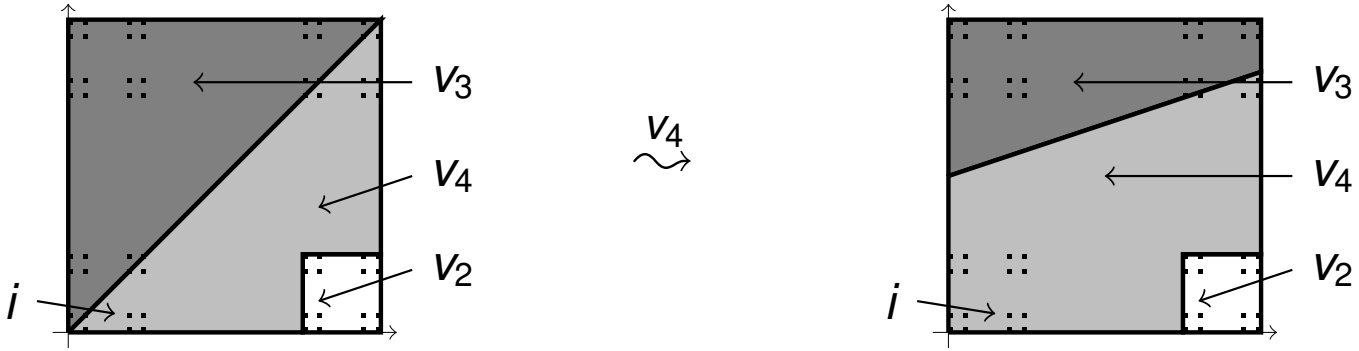


FineBlend: Training

- Incoming weights encode *responsible areas*:

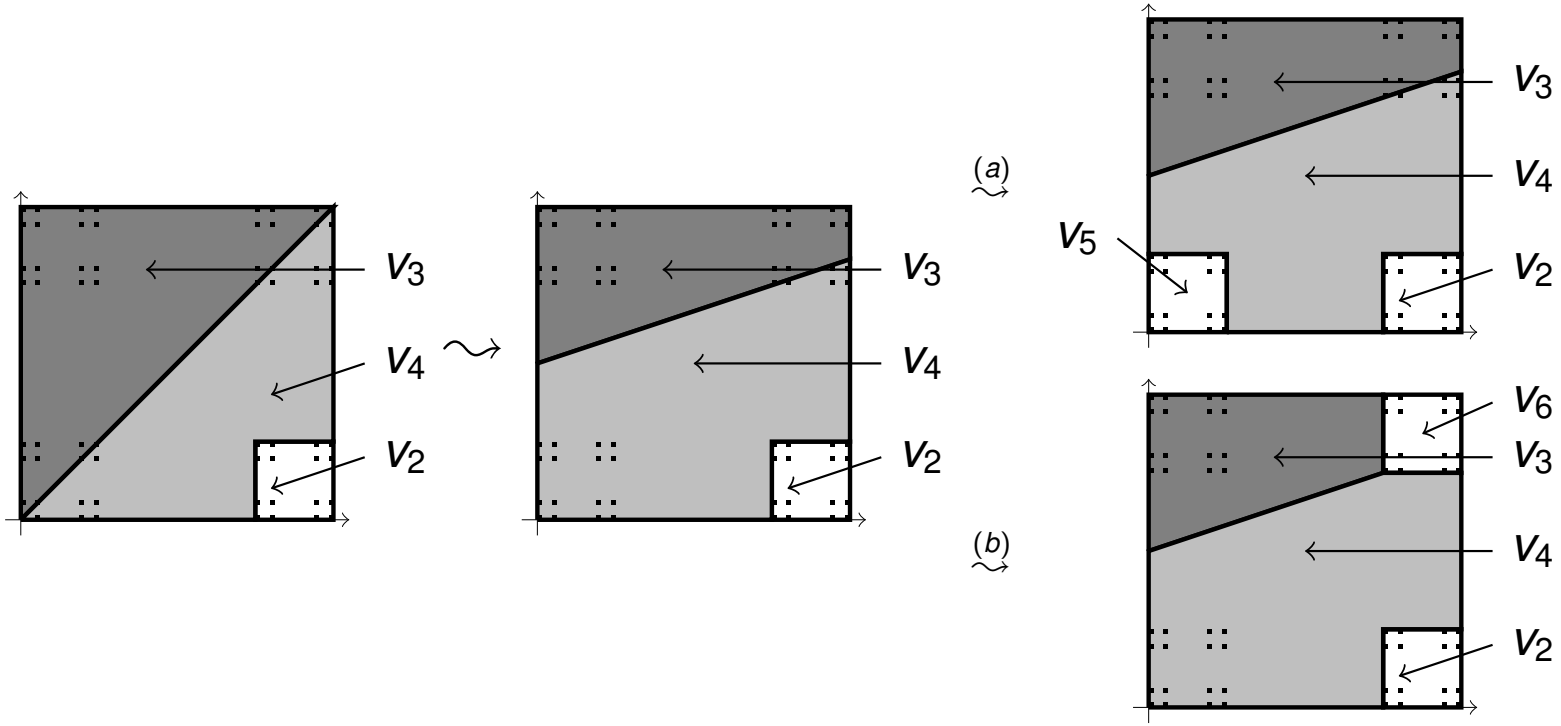


- Adapting the input weights:



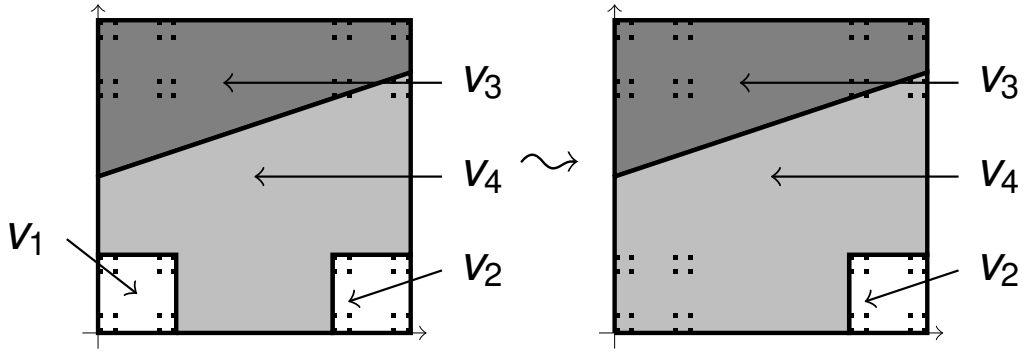
FineBlend: Adding New Units

- 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.

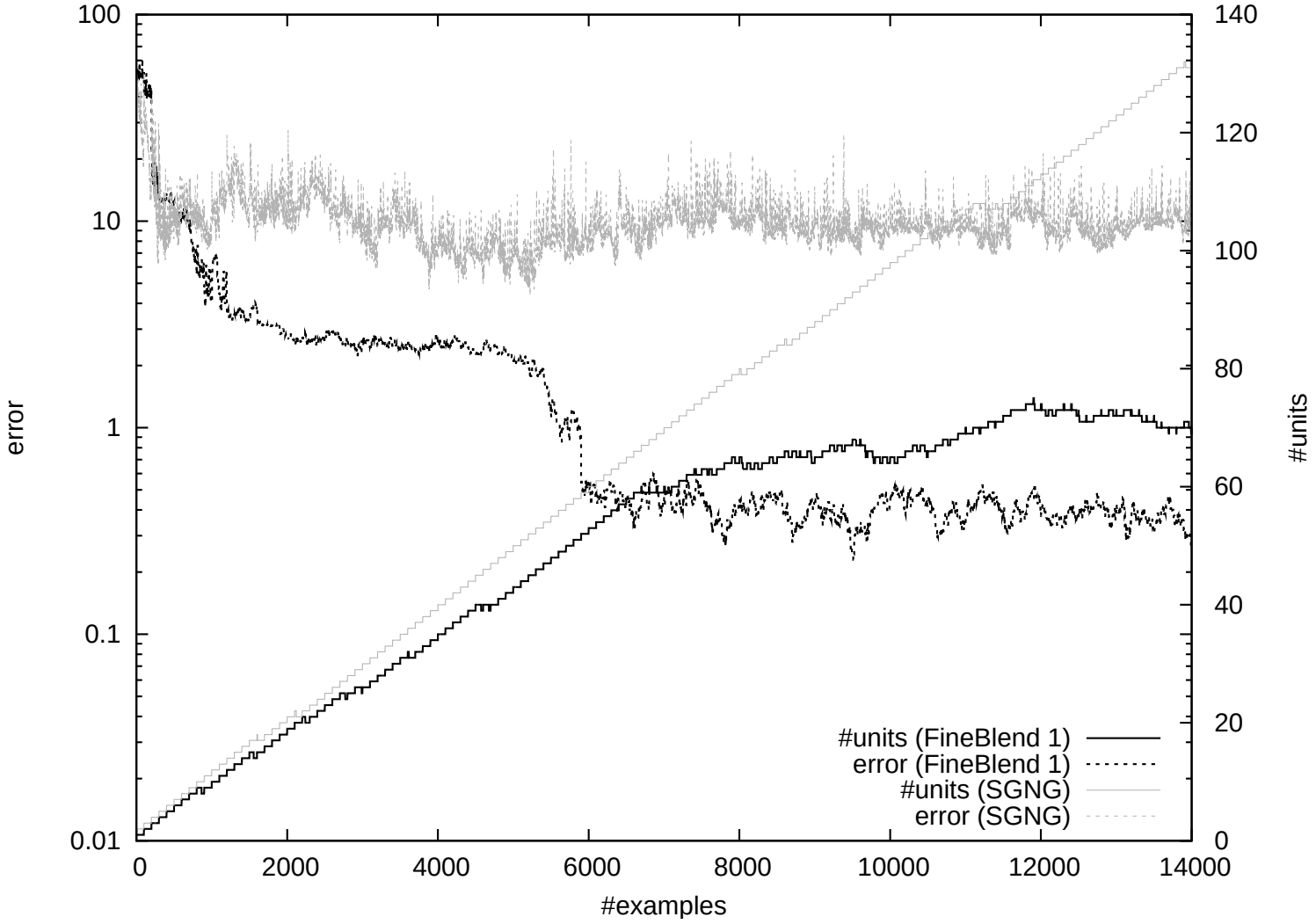


FineBlend: Removing Inutile Units

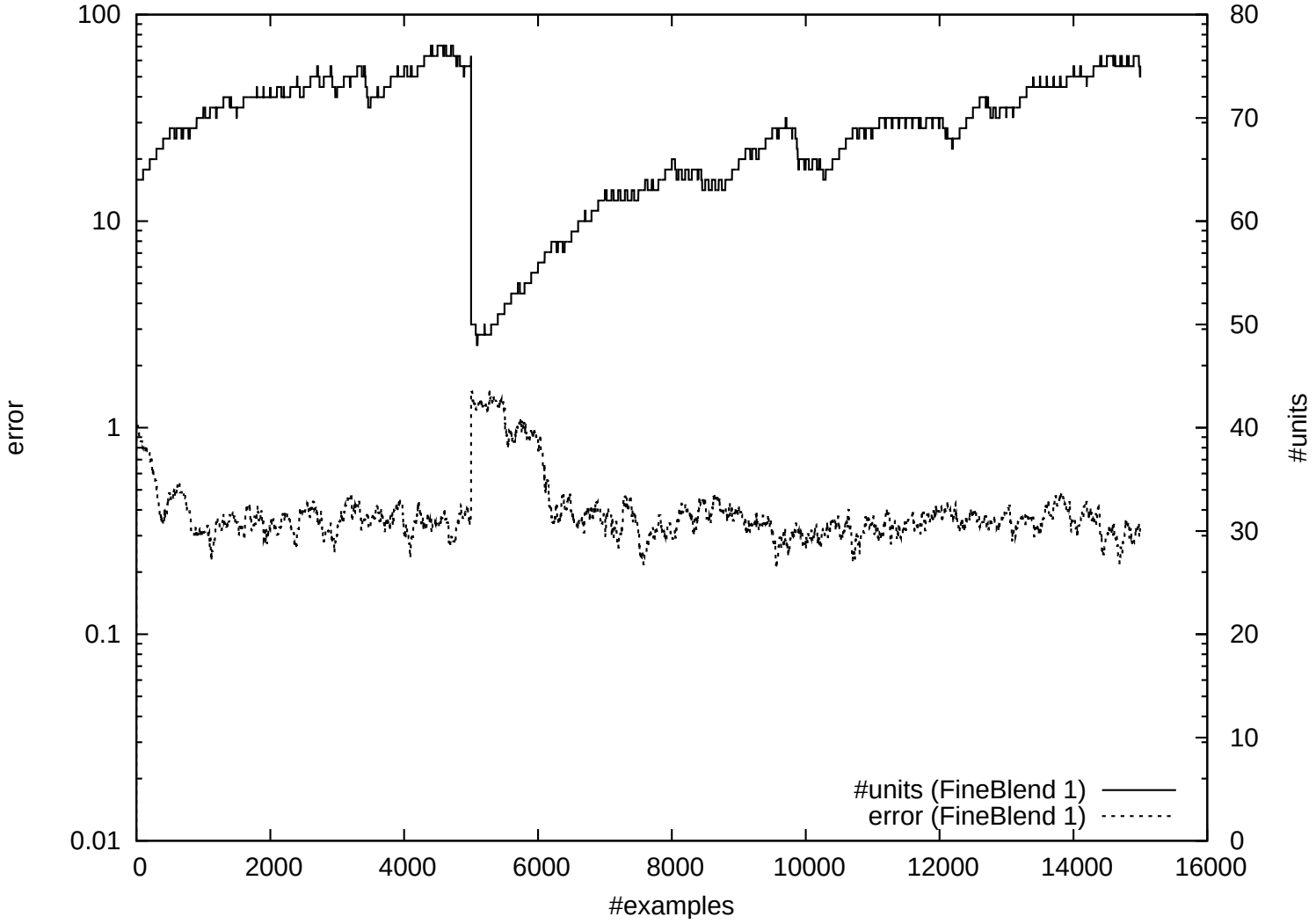
- Every unit maintains a utility value.
- If this drops below a given threshold, the corresponding unit is removed:



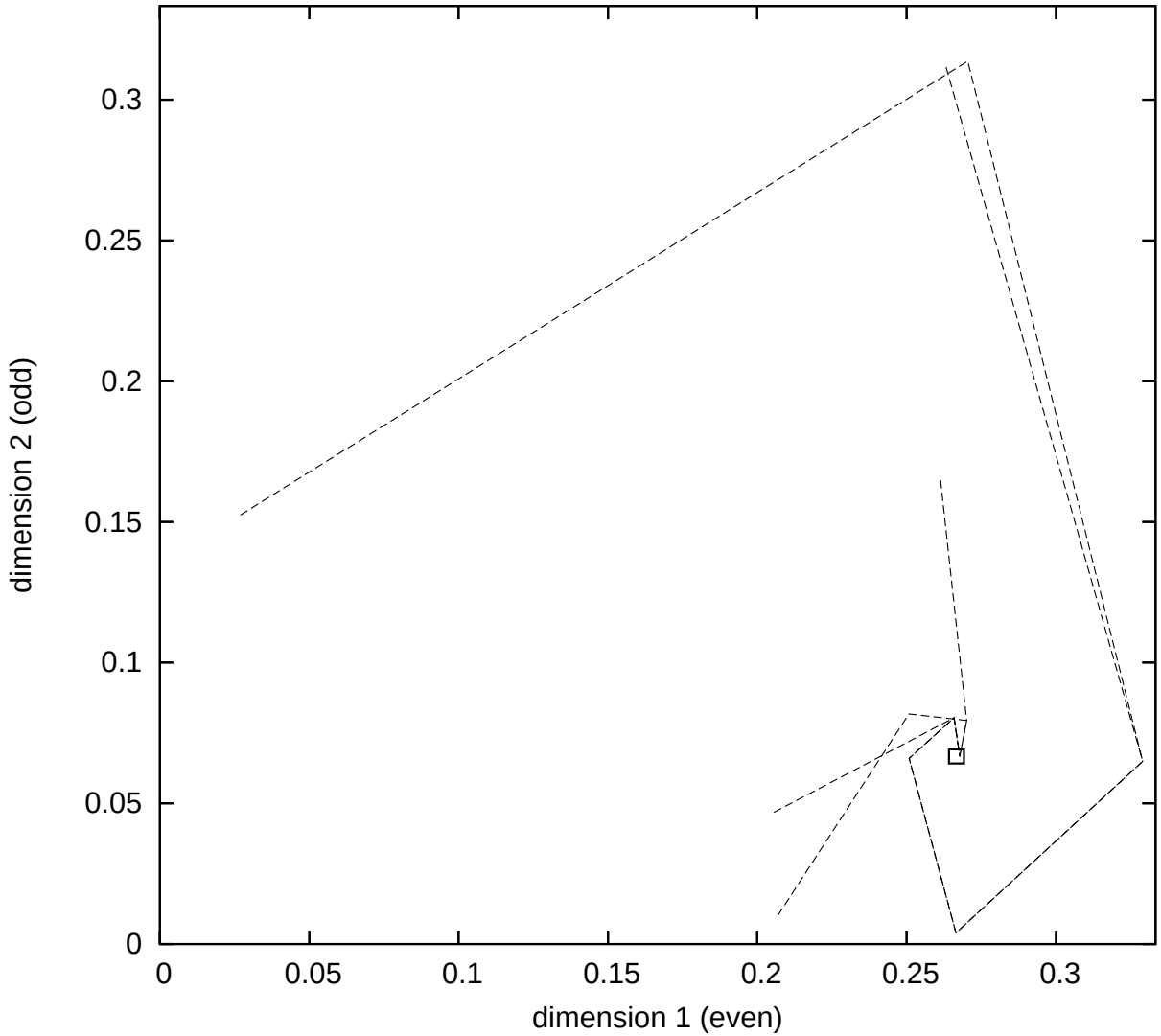
Statistics - FineBlend vs SGNG



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

First-order by propositional approximation

Let P be definite and I be its least Herbrand model (Seda & Lane, 2004):

- Choose some error ε.
- 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

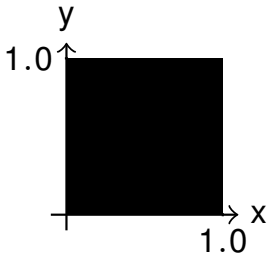
- Seda & Lane:
  - For definite programs under continuity constraint.
  - Treatment of acyclic programs should be ok.
  - Better approximation increases all layers of network.
  - Step functions only.
  - Sigmoidal approach (learning) to be investigated.
- Bader, Hitzler & Witzel:
  - For acyclic normal 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:

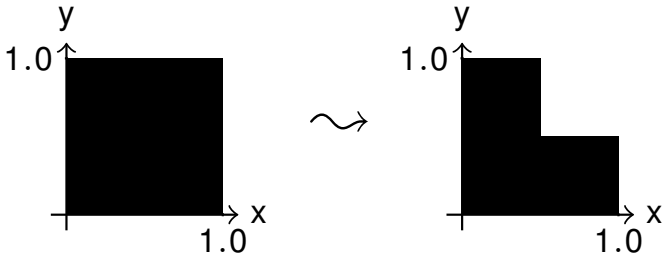
Iterated Function Symbols

- The Sierpinsky Triangle:



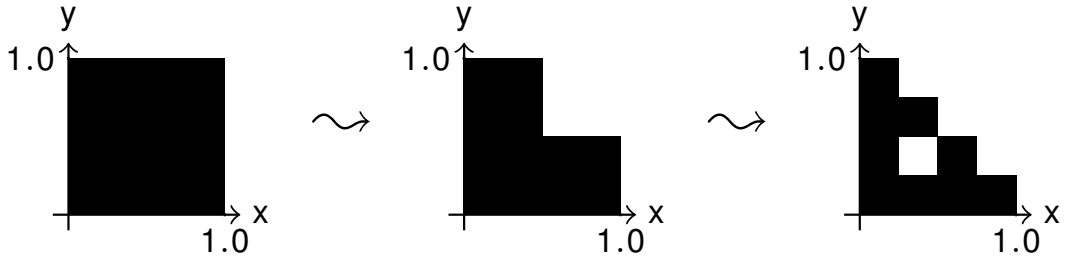
Iterated Function Symbols

- The Sierpinsky Triangle:



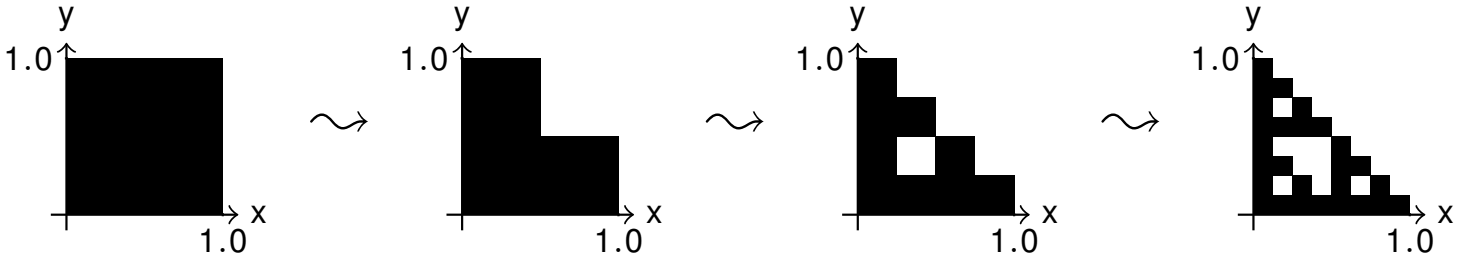
Iterated Function Symbols

- The Sierpinsky Triangle:



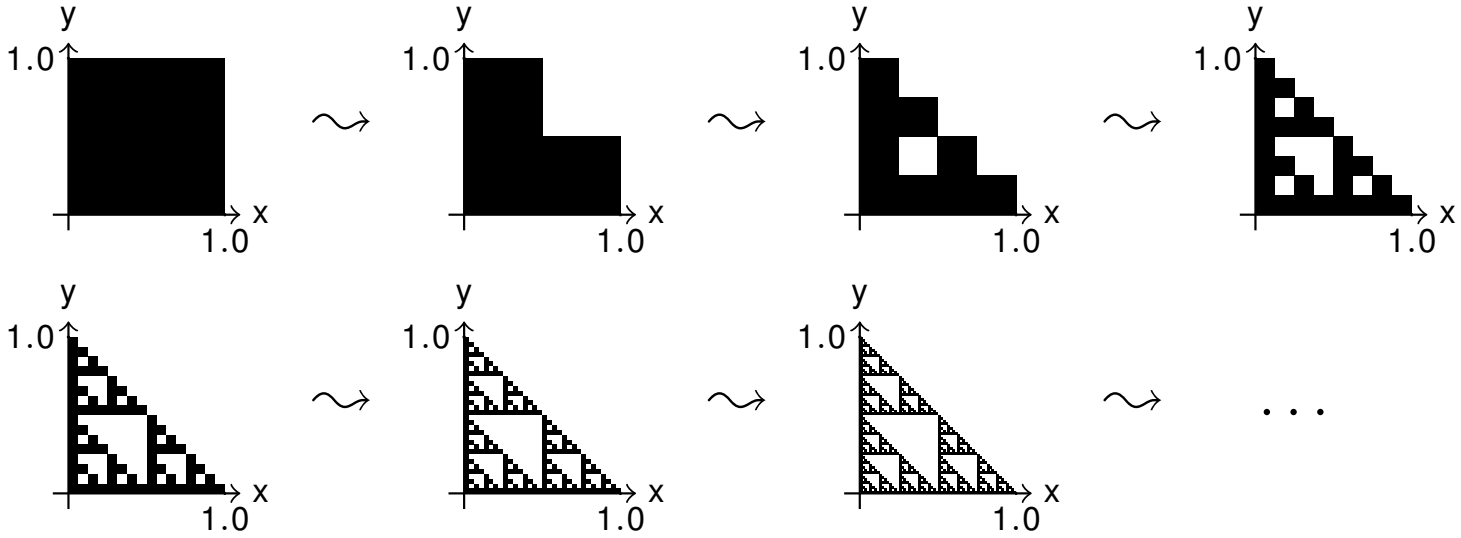
Iterated Function Symbols

- The Sierpinsky Triangle:



Iterated Function Symbols

- The Sierpinsky Triangle:



From Logic Programs to Iterated Function Systems

- For some logic programs we can explicitely 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-continous. 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

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 ther 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