Learning First-Order Rules with Differentiable Logic Program Semantics

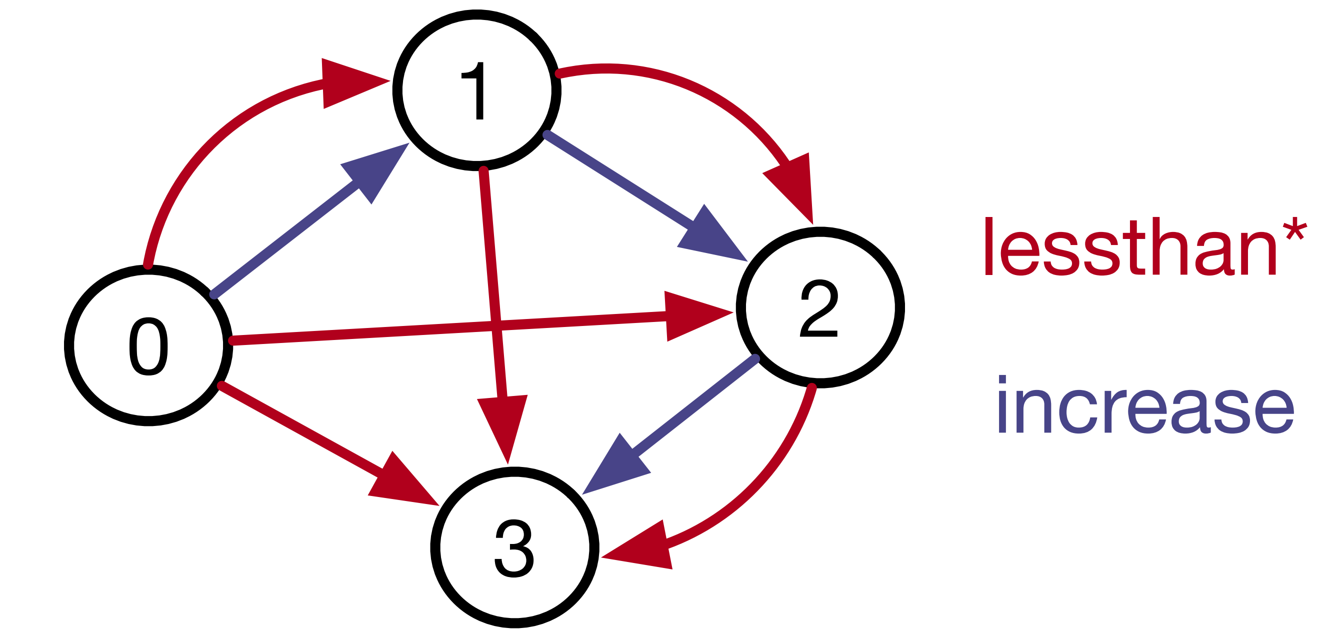
Kun Gao[1], Katsumi Inoue[2], Yongzhi Cao[1], Hanpin Wang[3, 1]
[1] Peking University
[2] National Institute of Informatics
[3] Guangzhou University

IJCAI ECAI VIENNA 22

北京大學 PEKING UNIVERSITY

NII 大学共同利用機関法人 情報・システム研究機構 国立情報学研究所 National Institute of Informatics

Motivation



$X \xrightarrow{P} Y \Leftrightarrow p(X,Y)$

0, 1, 2, 3	↔	constant
Lessthan, Increase	↔	predicate
X, Y	↔	variable

⇔

Background Knowledge

increase(0,1) *increase*(1,2) *increase*(2,3)

$\mathcal{B}$

Positive Example

lessthan(0,1) *lessthan*(1,2) *lessthan*(2,3)
lessthan(0,3) *lessthan*(0,2) . . .

$\mathscr{P}$

Negative Example

lessthan(1,0) *lessthan*(2,0) *lessthan*(3,1) . . .

$\mathscr{N}$

ILP ⇒

```
lt(X,Y) :- inc(X,Y).

lt(X,Y) :- inc(X,Z)&
           inc(Z,Y).
```

Solution

Preliminaries
Differential logic program semantics

- **Matrix representations for logic program**

  - P with the head atom $p(X, Y)$, 5 possible body atoms, and 3 different rules, and one of matrix embedding $\mathrm{M}_P \in [0,1]^{3\times 5}$ is:

$$
\begin{aligned}
p(X,Y) &\leftarrow p(Y,Z) \wedge p(Z,X).\\
p(X,Y) &\leftarrow p(X,Z) \wedge p(Z,Y).\\
p(X,Y) &\leftarrow p(Y,X).
\end{aligned}
$$

$$
\begin{array}{c}
\qquad\quad p(X, Z),\ p(Y,X),\ p(Y, Z),\ p(Z,X),\ p(Z, Y)\\
\mathbf{M}_P = \begin{bmatrix} 0 & 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}
\end{array}
\qquad \begin{bmatrix} 0 & 0 & \frac{1}{3} & \frac{2}{3} & 0 \end{bmatrix} \xRightarrow{\text{sum}} 1
$$

Preliminaries
Differential logic program semantics

- **Vector interpretations**:
  - An interpretation vector v ∈ $\{0,1\}^m$ represents an interpretation in a logic program. If the Boolean value of α_k is True, then $a_k = 1$; otherwise, $a_k = 0$

$$p(X,Y) \leftarrow p(Y,Z) \wedge p(Z,X).$$
$$p(X,Y) \leftarrow p(X,Z) \wedge p(Z,Y).$$
$$p(X,Y) \leftarrow p(Y,X).$$

Interpretation 1

$p(Y,Z),\ p(Z,X)$

Interpretation 2

$p(X,Z),\ p(Z,X)$

F_{body}= {p(X, Z), p(Y,X), p(Y, Z), p(Z,X), p(Z, Y)}

$\mathbf{v}_1 = [0,0,1,1,0]$

$\mathbf{v}_2 = [1,0,0,0,1]$

Preliminaries
Differential logic program semantics

Sakama et al., *ASPOCP*, 2018; Gao et al., *Mach. Learn.*, 2022;

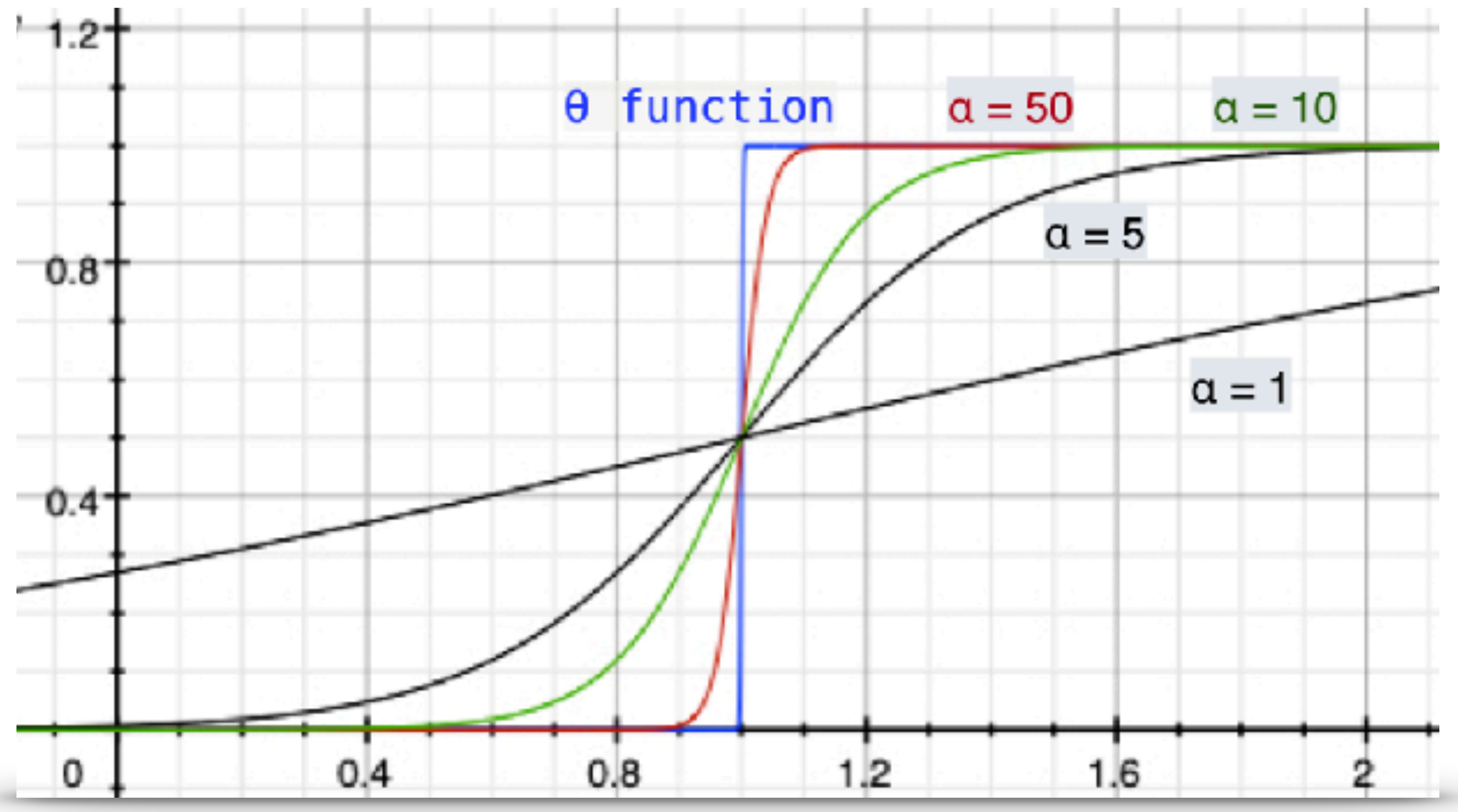
- **Deduction reasoning**:

$$\mathbf{v}_o = \vee_{k=1}^{m} \theta(\mathbf{M}_P[k, \cdot\,] \times \mathbf{v}_i^T)$$

$$\begin{array}{c} \begin{array}{ccccc} p(X, Z), & p(Y,X), & p(Y, Z), & p(Z,X), & p(Z, Y) \end{array} \\ \begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \times [0,0,1,1,0]^T\ (\mathbf{v}_i) = [1,0,0]^T\ (\mathbf{v}_o)$$

- **Differentiable deduction**:
  - Differentiable threshold function
  - Differentiable fuzzy logic (product-t norm)

$$\theta \;\Rightarrow\; \phi = \frac{1}{1 + e^{-\alpha(x-1)}} \qquad x \vee y \Rightarrow 1 - (1 - x)(1 - y)$$

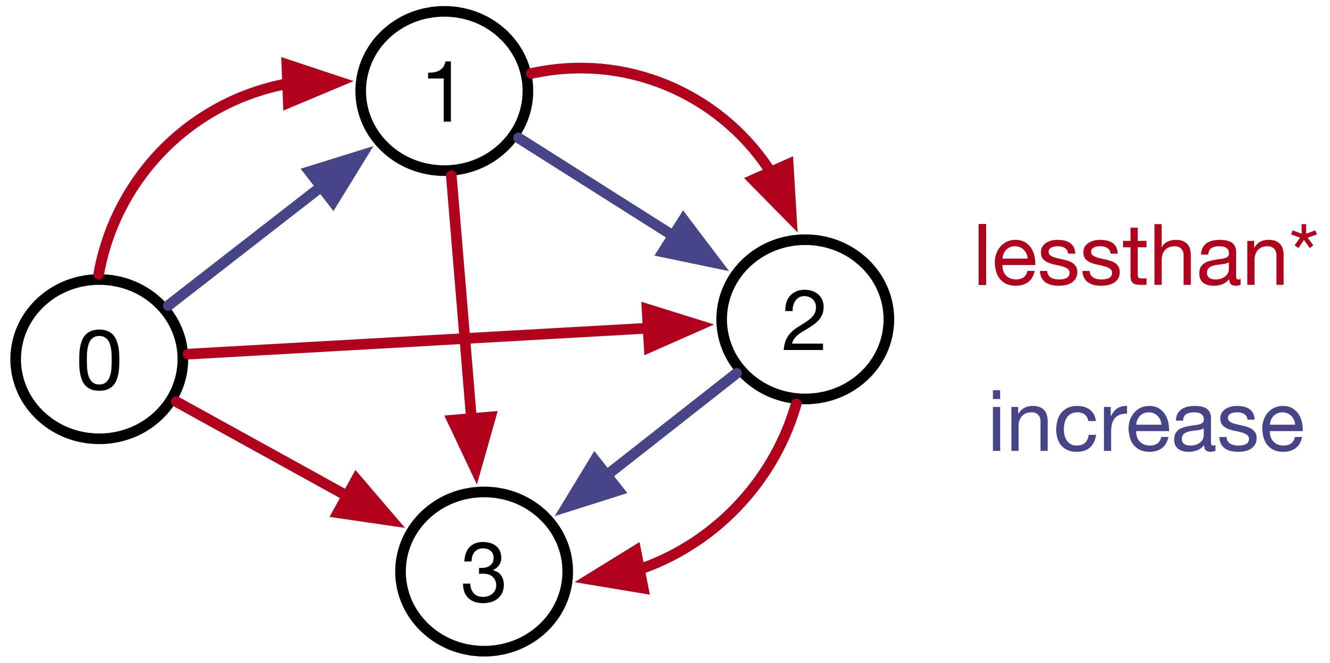


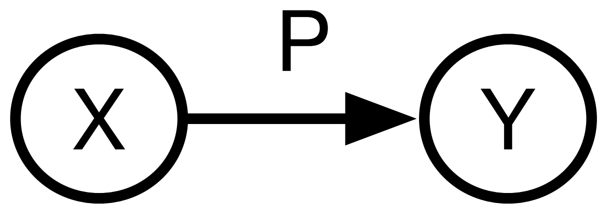
Methods

- **Propositionalization method**
  - Data preprocessing for relational facts
- **Differentiable inductive learning**
  - Learn parameters in matrix embeddings through neural networks
- **Logic rules extraction**
  - Interpret symbolic logic programs from matrix embeddings

Propositionalization

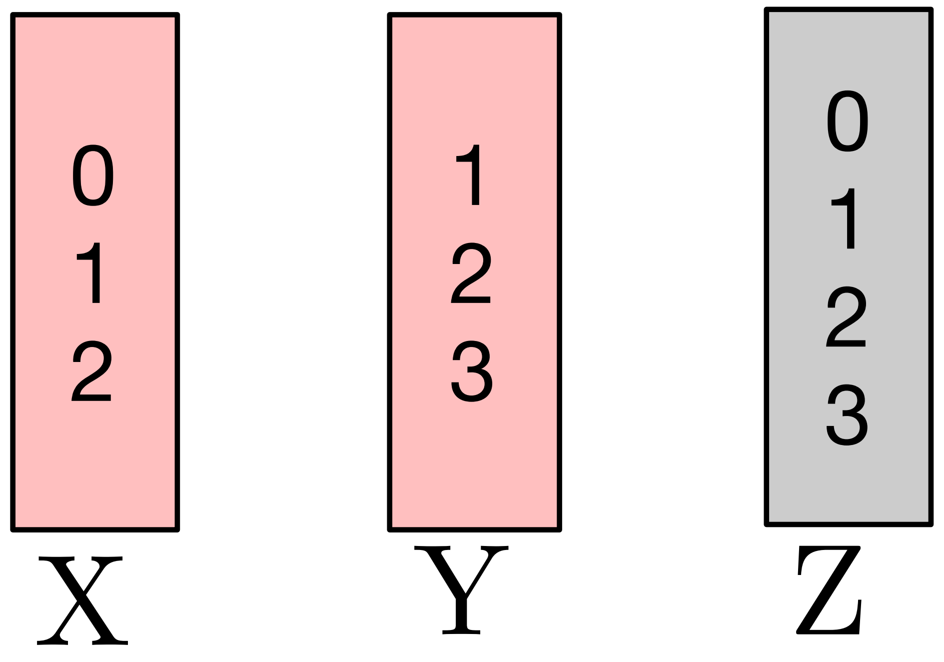
- Transfer the relational data into neural network readable attribute-valued data;
- **Input**: The training relational facts;
- **Output**: The trainable pairs of interpretation vectors;





$$\Leftrightarrow p(X, Y)$$

$$d = 1, V = \{X, Y, Z\}$$



1. **Preparation**: Allocate constants to variables

Propositionalization

$$\mathrm{X} \times \mathrm{Y} \times \mathrm{Z} = S$$

$$S = \{(0,1,0),,(0,1,1),\ldots,(2,3,3)\}$$

$$P_F = \{lt(X,Z), lt(Y,X), lt(Y,Z), lt(Z,X), lt(Z,Y),$$
$$inc(X,Y), inc(X,Z), inc(Y,X), inc(Y,Z), inc(Z,X), inc(Z,Y)\}$$

$$v_i = [0,0,0,0,1,1,0,0,0,0,1]$$
$$v_o = [1]$$
$$\vdots$$

2. **Generation**: Compute all substitutions and generate all input and output vector interpretation

$$v_i = [0,0,0,0,0,0,0,0,0,0,0] \quad v_o = [\_]$$

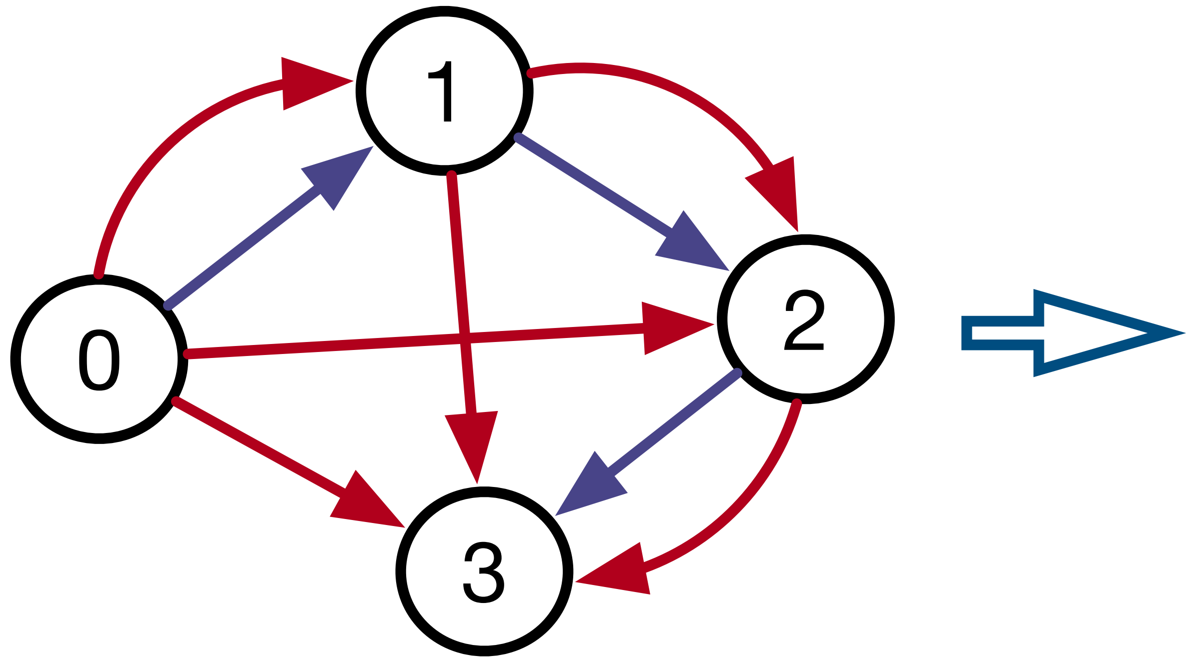
$$v_i = [0,\boxed{0},1,\boxed{0},1,1,0,0,0,\boxed{0},1]$$
$$v_i = [1,\boxed{0},0,\boxed{0},1,0,1,0,1,\boxed{0},1]$$
$$\vdots$$
$$v_i = [0,\boxed{0},0,\boxed{0},0,1,0,1,0,\boxed{0},1]$$

$$P_F = \{lt(X,Z), \boxed{lt(Y,X)}, lt(Y,Z), \boxed{lt(Z,X)}, lt(Z,Y),$$
$$inc(X,Y), inc(X,Z), inc(Y,X), inc(Y,Z), \boxed{inc(Z,X)}, inc(Z,Y)\}$$

3. **Examination**: Delete the noisy data and features

Propositionalization

- number of **valid** first-order features: $|v_i| = C$;
- $I_o = T_P(I_i)$ iff $v_o = D_P(v_i)$, where D_P is the differentiable deduction reasoning of a logic program P.



([0,0,0,1,1,0,0,0,1], [1])

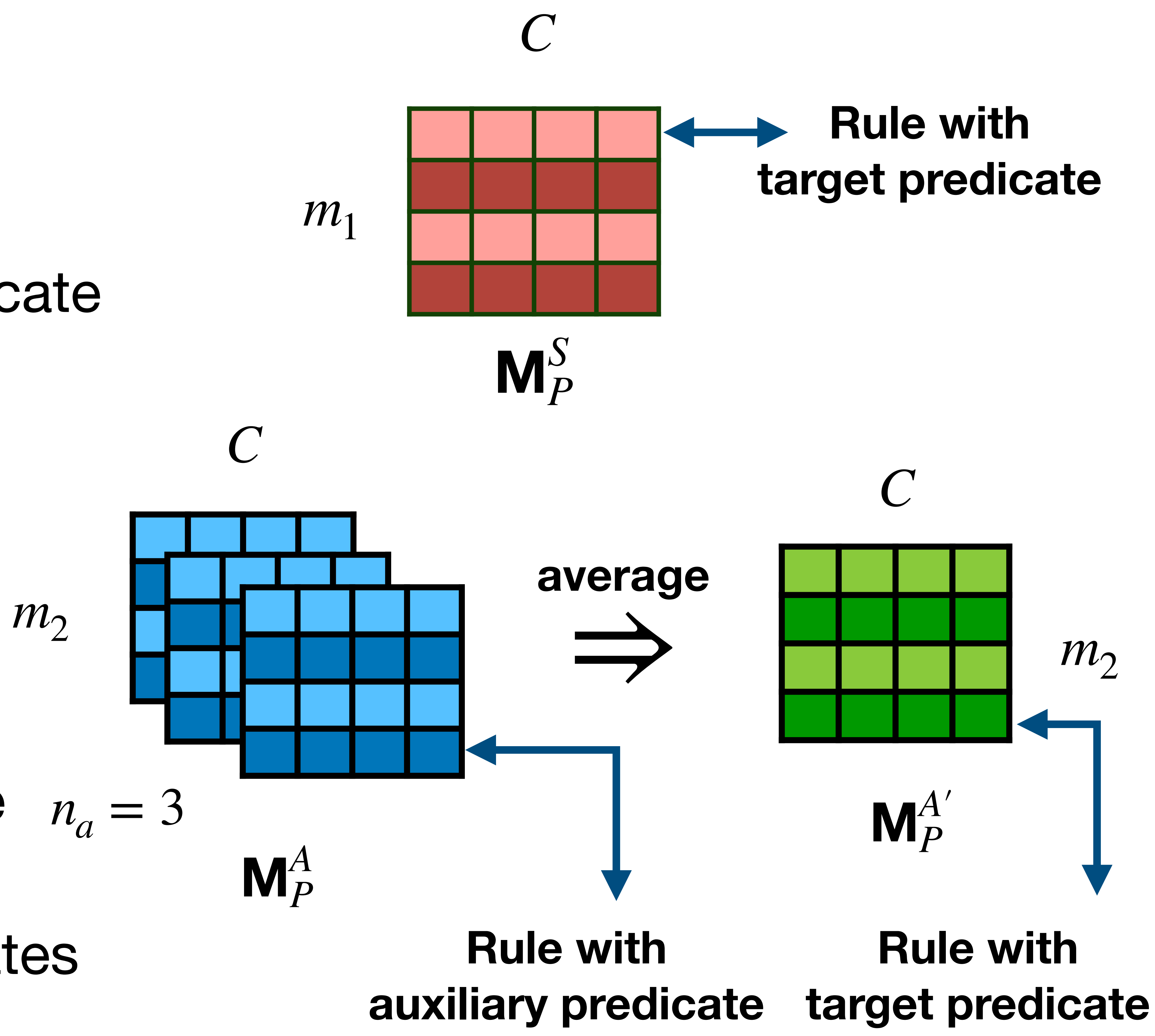
([0,0,0,0,1,0,1,0,0], [0])

⋮

([1,0,1,0,1,0,1,0,1], [1])

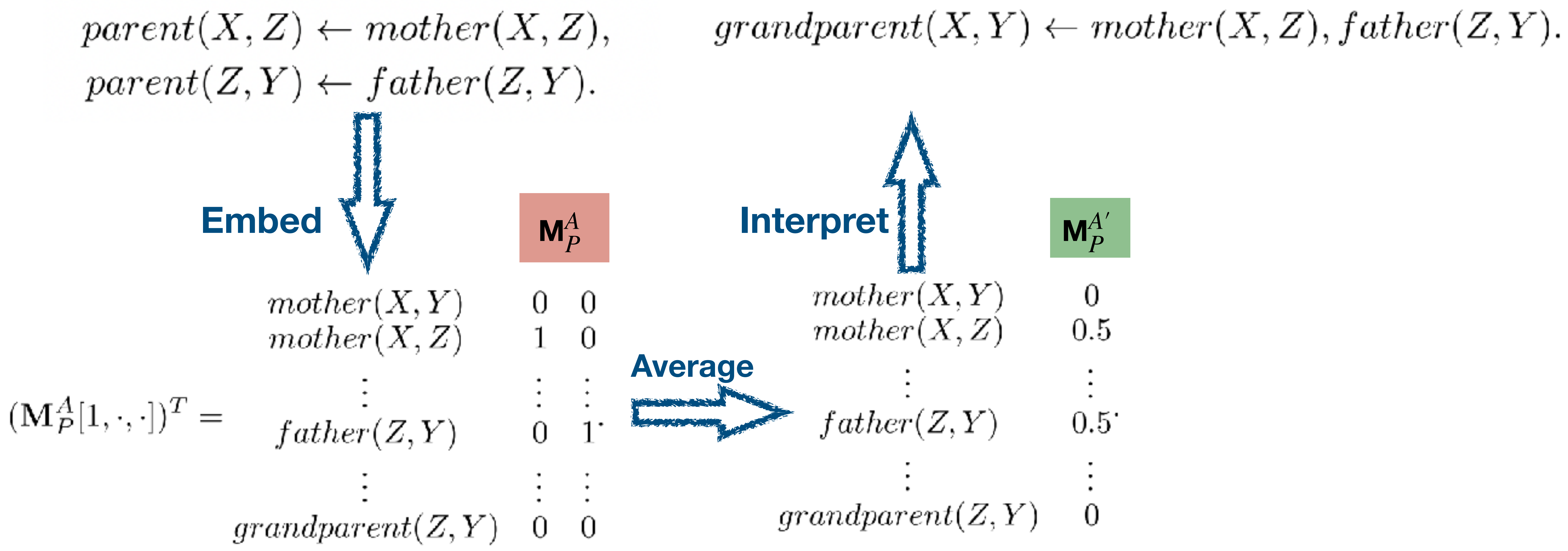
Differentiable Inductive Learning
Matrix Representation

- Trainable matrix $M_P^S \in [0,1]^{m_1 \times C}$
  - m_1 - the number of rules headed by target predicate
  - C - valid first-order features
- Trainable matrix $M_P^A \in [0,1]^{m_2 \times n_a \times C}$
  - m_2 - number of rules headed by target predicate
  - n_a - number of rules headed by auxiliary predicates



Differentiable Inductive Learning
Matrix Representation

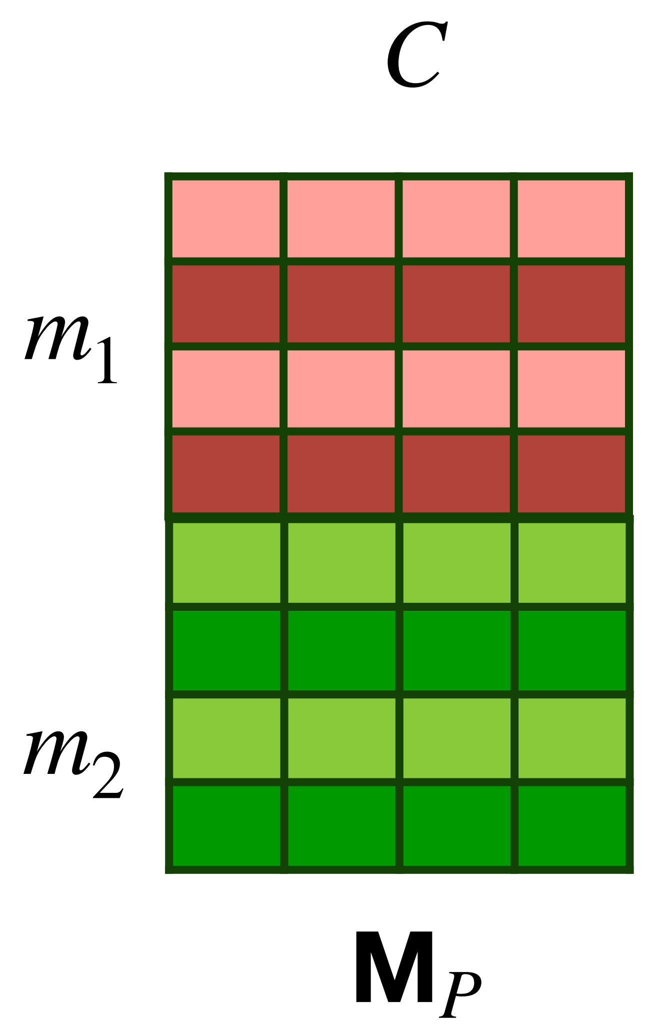
- The reason for applying average operation:
  - Keep the non-zero values index information from M_P^A to $\mathrm{M}_P^{A'}$
  - Increase the trainable parameters



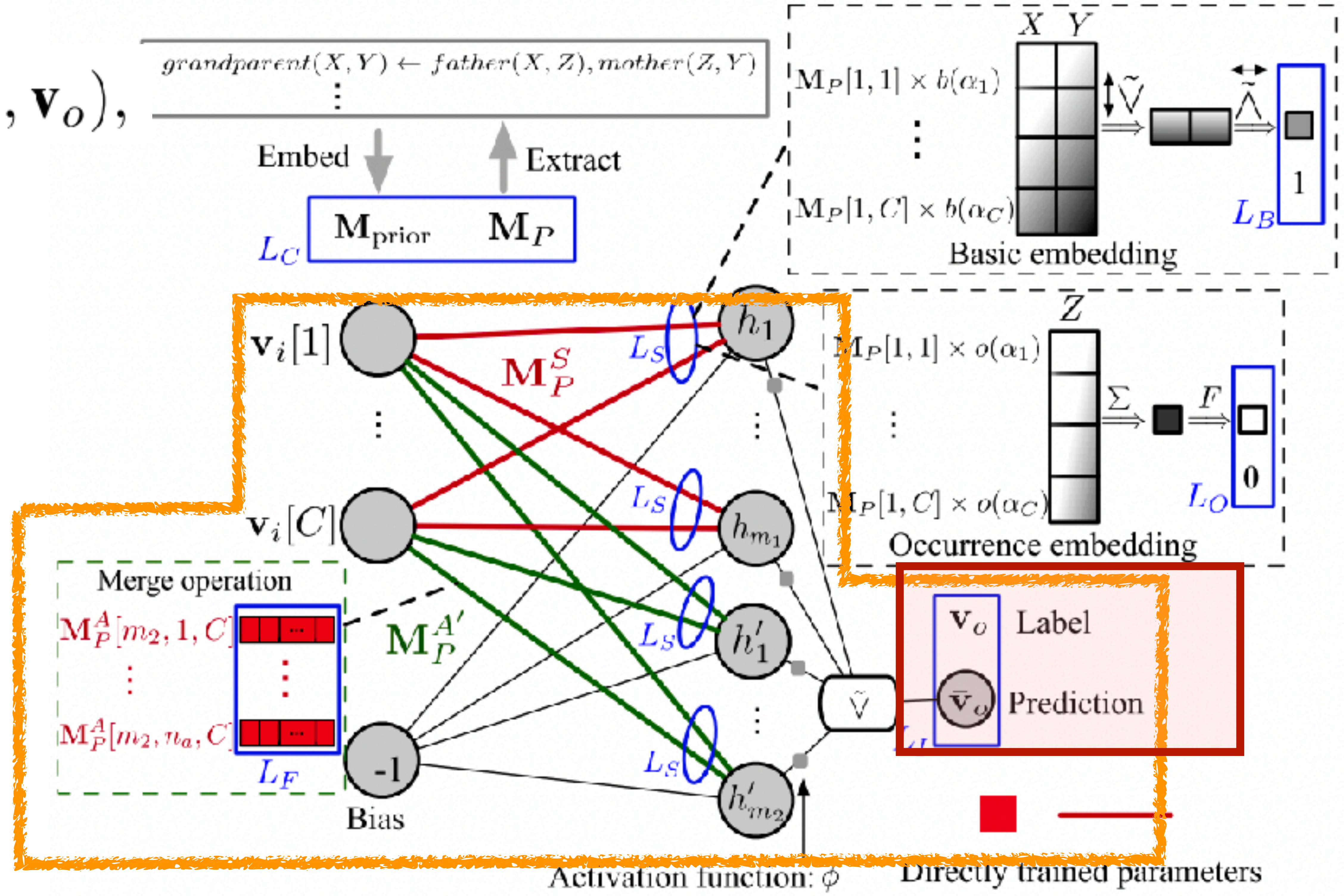
Differentiable Inductive Learning
Constraint Functions

- **Inference loss**:

$$\bar{\mathbf{v}}_o = \tilde{\triangledown}_{k=1}^{m}(\phi(\mathbf{M}_P[k,\cdot] \times \mathbf{v}_i^T - 1)),\ L_I = H(\bar{\mathbf{v}}_o, \mathbf{v}_o),$$



$\times [0,0,0,1,1,0,0,0,1] \Rightarrow v_o'$



Differentiable Inductive Learning
Forward-chained formats

- Generate rules with the forward-chained format:

$$p_t(X, Y) \leftarrow p_1(X, Z_1) \wedge p_2(Z_1, Z_2) \wedge \ldots \wedge p_{n+1}(Z_n, Y)$$

Head variables

Body variables

- **Basic constraint**: The body of each rule in an LP contains all variables that appear in the head atom.
- **Occurrence constraint**: In the body of each rule, the number of occurrences of each variable that does not appear in the header atom is not one.

Differentiable Inductive Learning
Forward-chained formats

- Generate rules with the forward-chained format:

$$n(Z_1) \geq 2$$

$$p_t(X, Y) \leftarrow p_1(X, Z_1) \wedge p_2(Z_1, Z_2) \wedge \ldots \wedge p_{n+1}(Z_n, Y)$$

- **Basic constraint**: The body of each rule in an LP contains all variables that appear in the head atom.

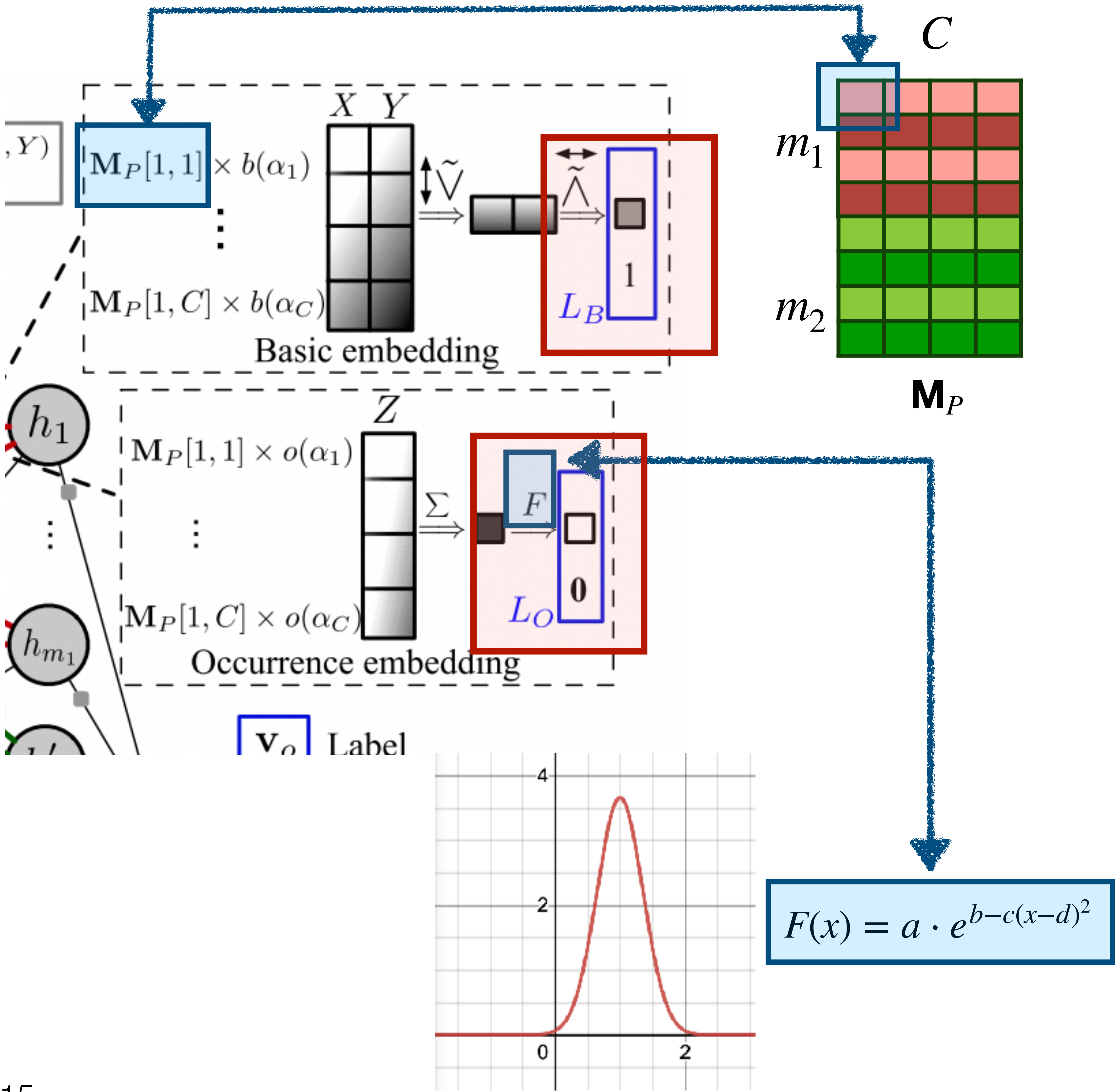
- **Occurrence constraint**: In the body of each rule, the number of occurrences of each variable that does not appear in the header atom is not one.

Differentiable Inductive Learning Constraint Functions

- Number of variables in head predicate a; Number of variable depth d.
  - **Basic embedding**: Atom -> $\{0,1\}^a$
  - **Occurrence embedding**: Atom -> $\{0,1\}^d$

$p_t(X, Y) \Rightarrow [1,1]$	$p_t(X, Y) \Rightarrow [0]$
$p_t(X, Z) \Rightarrow [1,0]$	$p_t(X, Z) \Rightarrow [1]$
$p_t(Z, Y) \Rightarrow [0,1]$	$p_t(Z, Y) \Rightarrow [1]$
Basic embeddings	Occurrence embeddings

  - **Basic Loss, Occurrence Loss**

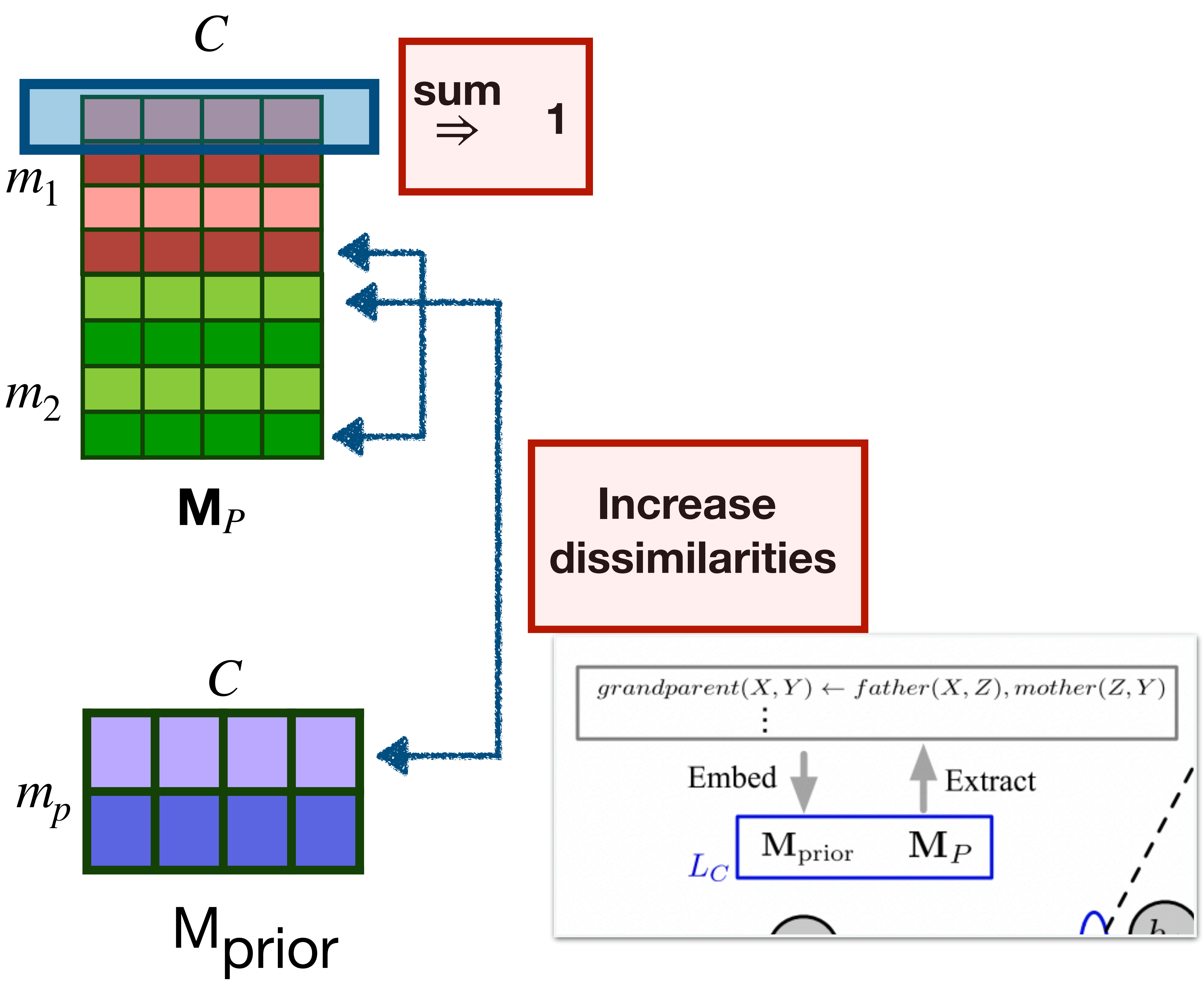


Differentiable Inductive Learning
Constraint Functions

- **Sum loss**:

$$L_S = \sum_{k=1}^{m} MSE(\sum_{i}^{C} \mathrm{M}_P[k,i],1)$$

- **Similar loss**:
  - Each row in M_P have more dissimilarities.
  - The rows in trainable matrix M_P and trained-well correct matrix $\mathrm{M}_{\text{prior}}$ have more dissimilarities.



Logical Rules Extraction

- Use a series of thresholds called τ_f to extract rules from M$_P$, and use Datalog to check the **precision** $\frac{n_r}{n_b}$ of a rule:
  - n_r - the number of substitutions meet both **head** and **body** of a rule
  - n_b - the number of substitutions meet only **body** of a rule
- Use τ_s to select the rules which precision values are larger than it.
- Use **accuracy (recall)** to indicate the ratio of the covered positive test examples.

$$
\begin{array}{cccccc}
 & \text{p(X, Z),} & \text{p(Y,X),} & \text{p(Y, Z),} & \text{p(Z,X),} & \text{p(Z, Y)} \\
\mathbf{M}_P = & \multicolumn{5}{c}{\begin{bmatrix} 0 & 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}}
\end{array}
$$

$\tau_f = \{0,0.2,\ldots,0.9\}$

$p(X,Y) \leftarrow p(Y,Z) \wedge p(Z,X).$	1.0
$p(X,Y) \leftarrow p(X,Z) \wedge p(Z,Y).$	1.0
$p(X,Y) \leftarrow p(Y,X).$	1.0
$p(X,Y) \leftarrow p(Z,X)$	0.2

$\tau_s = 0.5$

$p(X,Y) \leftarrow p(Y,Z) \wedge p(Z,X).$	1.0
$p(X,Y) \leftarrow p(X,Z) \wedge p(Z,Y).$	1.0
$p(X,Y) \leftarrow p(Y,X).$	1.0

Experimental Results

- From ILP datasets, the generated rules have:
  - $\tau_s = 1$, so the precision values are 1.
  - Accuracy value is 100%.

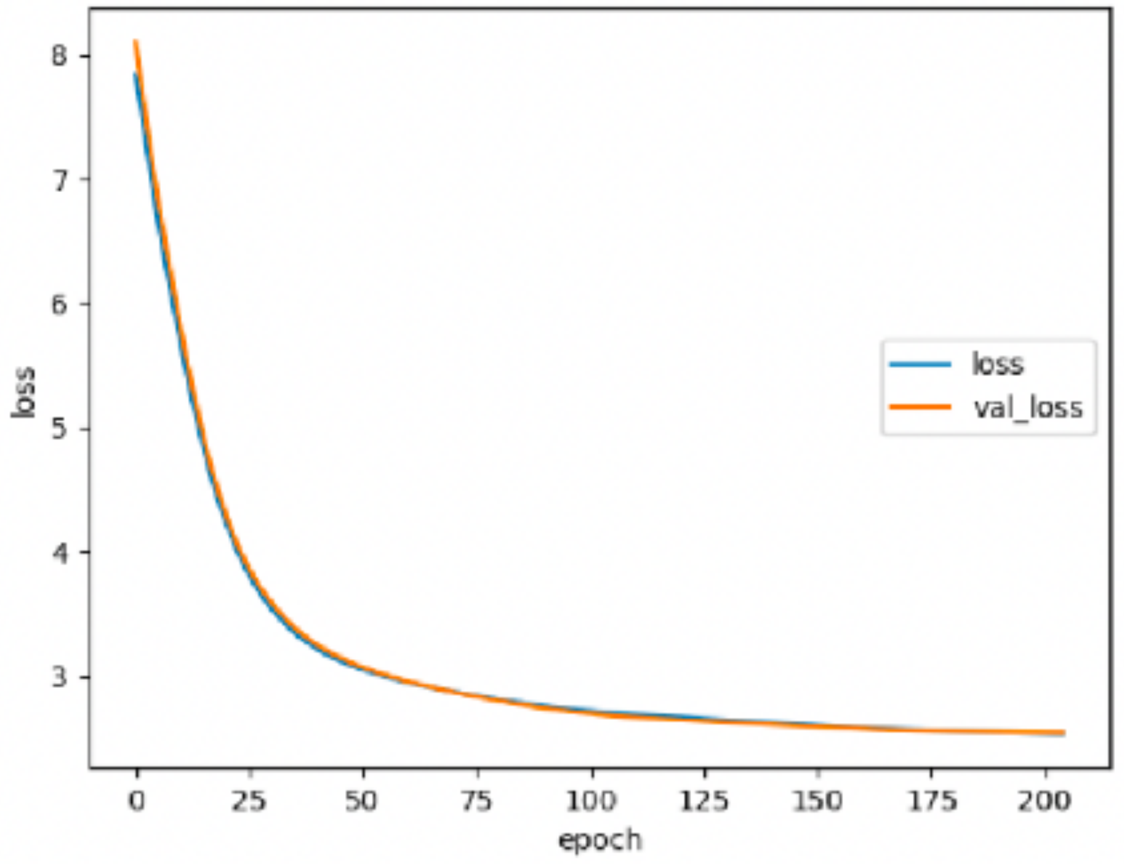


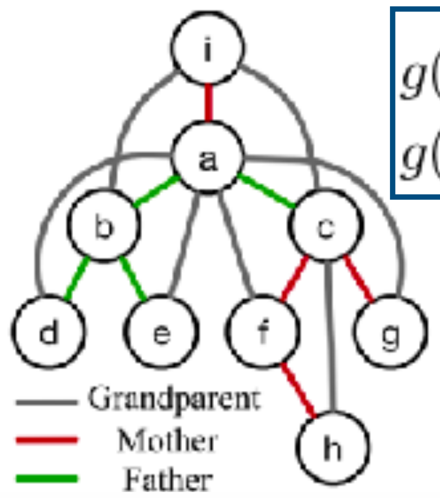
Figure 6: Learning curve in the *lessthan* task.

```
1) even(X) :- zero(X).
2) even(X) :- succ(X,Z)& succ(Z,Y)& even(Y).
```

The program in **Even** dataset

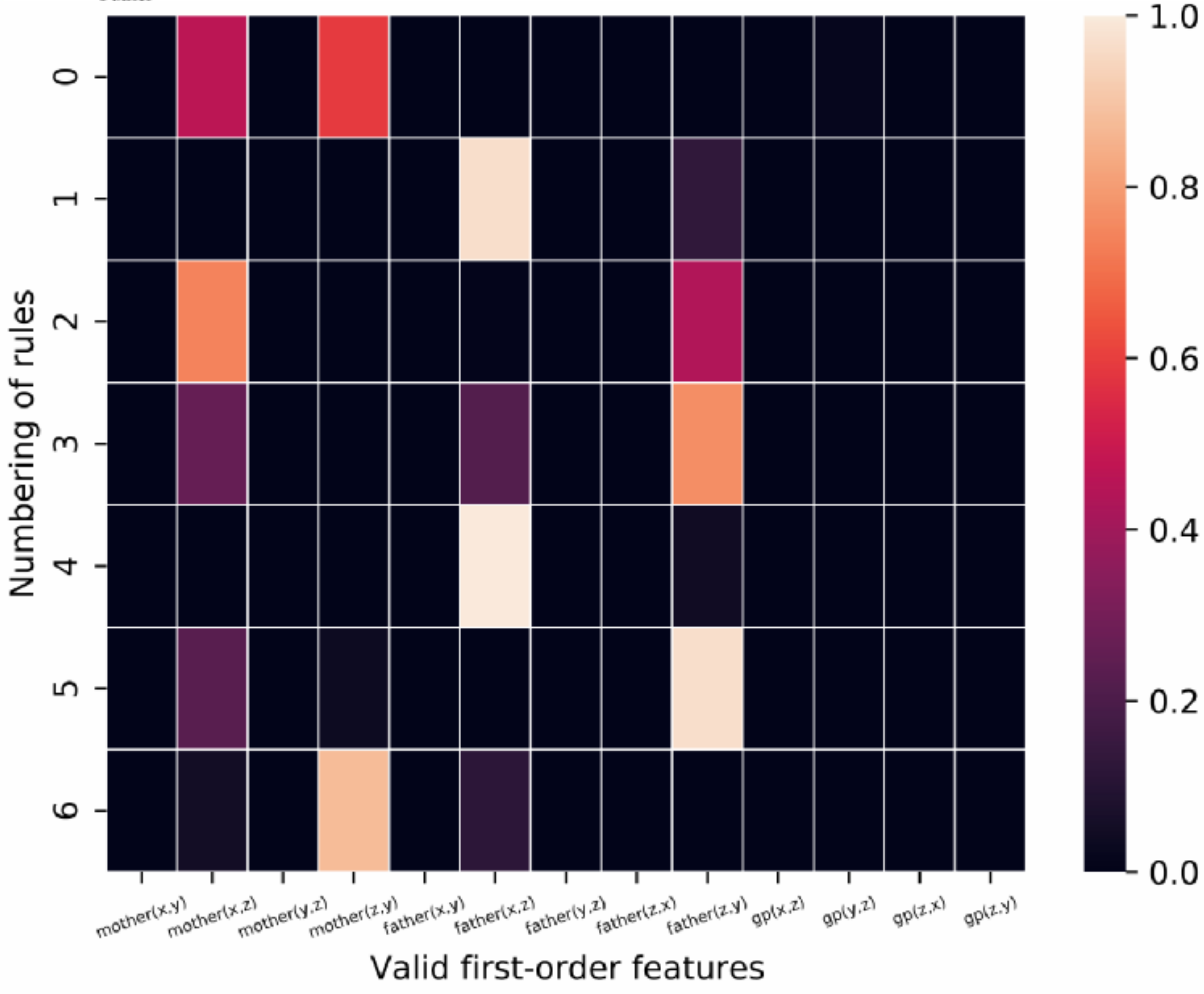
Table 4: The results on ILP datasets. The symbols ✓, $*$, and — indicate that the accuracy of the generated logic program is equal to 100%, less than 100%, and equal to 0%, respectively.

Domain	Task	∂ILP	NeuralLP	DFOL
Arithmetic	Predecessor	✓	✓	✓
	Odd	✓	—	✓
	Even / Succ2 (10)	✓	—	✓
	Even / Succ2 (20)	—	—	✓
	Lessthan	✓	✓	✓
	Fizz	✓	—	✓
	Buzz	✓	—	✓
Lists	Member	✓	*	✓
	Length	✓	—	✓
Family Tree	Son	✓	*	✓
	Grandparent	✓	✓	✓
	Husband	✓	—	—
	Uncle	✓	—	—
	Relatedness	✓	*	✓
	Father	✓	—	✓
Graphs	Directed Edge	✓	*	✓
	Adjacent to Red	✓	—	✓
	Two Children	✓	—	✓
	Graph Coloring (6)	✓	—	✓
	Graph Coloring (10)	—	—	✓
	Connectedness	✓	*	✓
	Cyclic	✓	—	✓



$$g(X,Y) \leftarrow m(X,Z) \wedge m(Z,Y),\ g(X,Y) \leftarrow f(X,Z) \wedge f(Z,Y)$$
$$g(X,Y) \leftarrow m(X,Z) \wedge f(Z,Y),\ g(X,Y) \leftarrow f(X,Z) \wedge m(Z,Y)$$

The program in **Grandparents** dataset



The learned matrix $\mathbf{M}_P$

Experimental Results

- From fuzzy data, we let $\varepsilon \sim N(0, \sigma^2)$
    - $p_i^+ = min(1 - \varepsilon, 1)$
    - $p_i^- = max(\varepsilon, 0)$
- For the **mislabeled data**, both positive and negative training examples are mislabeled:
    - μ ranging from 0.05 to 1 with step 0.05.

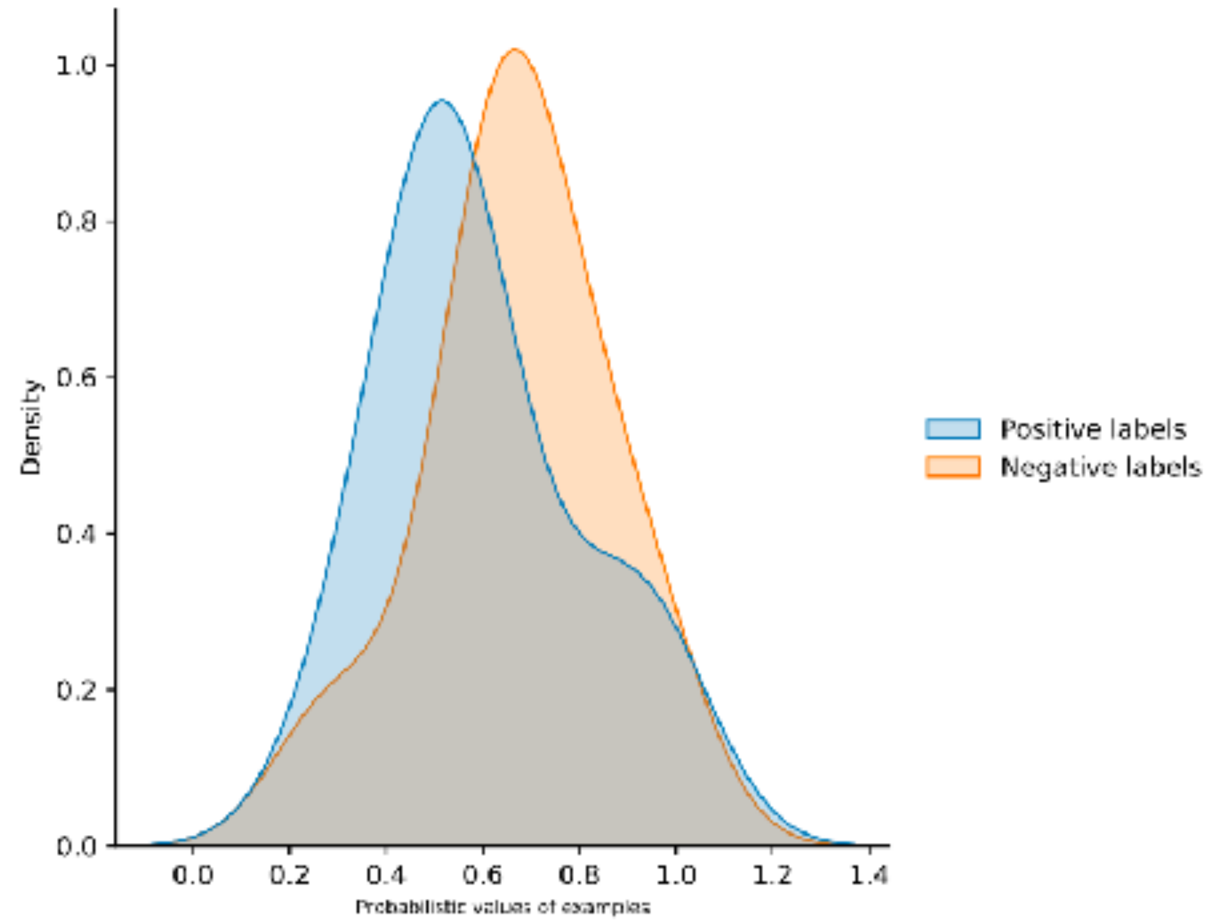


Figure 4: The distribution of examples in the *lessthan* task when $\sigma = 3$.

Table 1: The results on ambiguous datasets. The notations *Con* and *DE* represent *connectedness* and *directed edge* tasks, respectively.

	Lt	*Pre*	*Member*	*Son*	*Con*	*DE*
σ	3	3	3	3	2	3
μ	0.95	0.95	0.90	0.95	0.95	0.95

When $\tau_s = 1$ and accuracy is 100%, the largest standard deviation σ and mislabeling rate μ

Experimental Results

- From knowledge graph datasets,
    - Set $\tau_s = 0.3$;
    - Check the HIT@n and MRR on three knowledge graph datasets.

Table 3: The descriptions of the knowledge bases.

Dataset	#Object	#Relation	#Fact
Countries	252	2	1158
Nations	14	56	2565
UMLS	135	49	6529

```
1) locatedIn(X,Y) :- locatedIn(Z,Y)& neighborOf(X,Z)& neighborOf(Z,X). #(0.68, 714, 1053)
2) locatedIn(X,Y) :- locatedIn(Z,Y)& neighborOf(X,Z). #(0.68, 723, 1067)
3) locatedIn(X,Y) :- locatedIn(X,Z)& locatedIn(Z,Y). #(0.81, 187, 231)
4) locatedIn(X,Y) :- locatedIn(Z,Y)& neighborOf(Z,X). #(0.68, 723, 1068)
```

The program in **Country**-S1 dataset

Table 2: Comparison on knowledge bases. The results in bold indicate the highest accuracy on the corresponding test datasets. The ACC@Sn represent the accuracy of the generated LP on the Sn subset of Countries. The ACC@h represent the accuracy of LP with the head predicate h. Besides, *blo*, *int*, *neg*, and *intw* denote the relations of *blockpositionindex*, *intergovorgs3*, *negativecomm*, and *interacts_with*.

Dataset	Metrics	NTPλ	NeuralLP	DFOL
Countries	ACC@S1	**100.00**	**100.00**	**100.00**
	ACC@S2	**100.00**	**100.00**	**100.00**
	ACC@S3	**100.00**	–	–
Nations	MRR	41.79	56.49	**78.88**
	HITS@1	41.79	52.49	**73.88**
	HITS@3	41.79	60.95	**84.58**
	HITS@10	41.79	61.19	**85.07**
	ACC@*blo*	**100.00**	50.00	**100.00**
	ACC@*int*	**84.62**	**84.62**	**84.62**
	ACC@*neg*	37.50	**75.00**	**75.00**
UMLS	MRR	30.03	66.69	**74.96**
	HITS@1	29.95	61.27	**71.41**
	HITS@3	30.11	72.31	**78.82**
	HITS@10	30.11	72.31	**78.97**
	ACC@*isa*	65.96	63.83	**91.48**
	ACC@*intw*	83.67	86.67	**100.00**

Conclusion and Future Work

- Conclusion
  - DFOL generates first-order logic programs usually with variable depth 1 or 2;
  - DFOL is a fast, precise, robust, scalable rule learner.
- Future Work
  - Larger variable depth
  - Support negation and function in logic programs

Thanks for your attention！