PEKING UNIVERSITY

Learning from Interpretation Transitions Using Differentiable Logic Programming Semantics

Kun Gao[1], Hanpin Wang[1, 2], Yongzhi Cao[1], Katsumi Inoue[3]

[1] Peking University

[2] Guangzhou University

[3] National Institute of Informatics

2021/10/25

Background knowledge

- A normal logic program is a finite set of rules that satisfies the following form:

$$h \leftarrow l_1 \wedge l_2 \wedge \cdots \wedge l_n,$$

  where the l_i's are **literals** and h is the head atom. Given a logic program P, the set of all atoms (ground atoms) in P is called a **Herbrand base** and is denoted as $\boldsymbol{B_P}$.

- An interpretation of a logic program is a subset of B_P. Given a normal logic program P and $B_P = \{p_1, \ldots, p_n\}$, an interpretation vector is denoted as $\boldsymbol{v} = (v_1, \ldots, v_n)^T \in \{0, 1\}^n$.

Introduction: the definition of learning task

- Given a propositional logic program P, the immediate consequence operator $T_P : 2^{B_P} \rightarrow 2^{B_P}$ is defined as follows:

$$T_P(I) = \{head(r) \mid r \in P, body^{+}(r) \subseteq I, body^{-}(r) \cap I = 0\}$$

The definition of the learning task

- Input: the pairs of interpretation transitions (I, J), where $\boldsymbol{J = T_P(I)}$.
- Output: the logic program P.

Learning steps

Model: Learning normal matrices

Normal matrices

Given a normal logic program, we use **normal (NOR) matrix** to represent it. The normal matrix of a logic program $\mathbf{M}_P^{NOR}$ and the pairs of interpretation vectors meet: $\boldsymbol{v_o} = \theta(\mathbf{M}_P^{NOR}\boldsymbol{v_i})$, where the threshold is defined as follows:

$$\theta(x) = \begin{cases} 1 & \text{if } x \geq 1; \\ 0 & \text{otherwise.} \end{cases}$$

The denominator of the fraction, corresponding the literal *p,* is the number of literals in the conjunctive clause with the least literals that appear in the corresponding rule

$$P: \begin{array}{l} p \leftarrow (p \wedge q \wedge \neg r) \vee (p \wedge \neg q) \\ q \leftarrow p \wedge r \\ r \leftarrow \end{array} \quad \Longrightarrow \quad \mathbf{M}_P^{NOR} = \begin{array}{c|cccccc} & p & q & r & \neg p & \neg q & \neg r \\ \hline (p) & \frac{1}{2} & \frac{1}{3} & 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ (q) & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ (r) & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

NOR matrix

Model: Learning normal matrices

We use a differentiable function ϕ to replace the function θ:

$$\phi = \frac{1}{1 + e^{-\alpha x}}$$

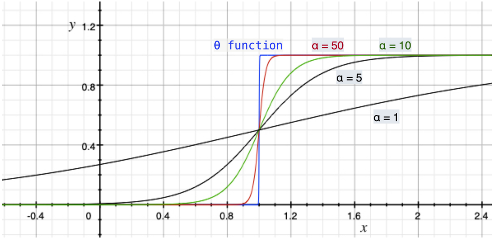


Figure: The curves of the function θ and $\phi(x-1)$

Model: Learning normal matrices

- We define a differentiable formula to represent the immediate consequence function T_P and the loss function:

$$\overline{\mathbf{v}}_o = \phi(\overline{\mathbf{M}}_P^{NOR}\mathbf{v}_i - \mathbf{1}),$$

$$\text{loss} = \lambda_1 \cdot \boxed{H(\mathbf{v}, \overline{\mathbf{v}})} + \lambda_2 \cdot \boxed{\sum_{c=1}^{m}\sum_{b=1}^{2n} \overline{\mathbf{M}}_P^{NOR}[c, b],}$$

H is the cross-entropy function

penalty term

Model: Connection of two learner

Meta-info learner

- Extract the number of non zero elements in every line, denoted as $\boldsymbol{l_{p_k}}$, from the generated normal matrices.

$$
\mathbf{M}_P^{NOR} = \begin{array}{c} \\ (p) \\ (q) \\ (r) \end{array} \begin{array}{cccccc} p & q & r & \neg p & \neg q & \neg r \\ \frac{1}{2} & \frac{1}{3} & 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array} \begin{array}{c} \\ \Rightarrow \\ \Rightarrow \\ \Rightarrow \end{array}
$$

NOR matrix

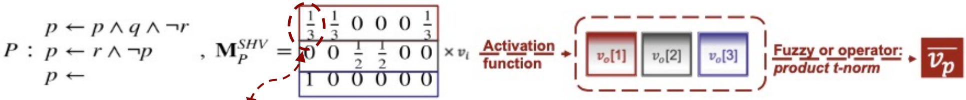
The *meta-information*: $l_p = 4,\ l_q = 2,\ l_r = 1.$

Interpretation learner

- The number of rules m in a same head variable logic program is no larger than $C\left(l_{p_k}, \left\lfloor \frac{l_{p_k}}{2} \right\rfloor\right)$.
- For all $k \in [1, C\left(l_{p_k}, \left\lfloor \frac{l_{p_k}}{2} \right\rfloor\right)]$, learn the **same head variable (SHV)** matrix representing the k-th rule in the disjunctive normal form logic program.

Model: Learning SHV matrices

- We set multiple trainable matrices M_P^{SHV} to represent the different disjunctive normal form rules in the logic program P.



The denominator of the fraction, corresponding the literal *p,* is the number of literals in the corresponding rule

$$\bar{\mathbf{v}}_o[k] = 1 - \prod_{c=1}^{C(L_{p_k}, \lfloor \frac{L_{p_k}}{2} \rfloor)} \left(1 - \phi\left(\sum_{b=1}^{2n} \overline{\mathbf{M}}_{P_k}^{SHV}[c, b] \cdot \mathbf{v}_i[b] - 1\right)\right), \ 1 \leq k \leq m.$$

product t-norm

The loss function is:

$$\text{loss} = \sum_{k=1}^{m}(\lambda_1 \cdot H(\mathbf{v}[k], \overline{\mathbf{v}}[k]) + \lambda_2 \cdot \sum_{c=1}^{C(L_{p_k}, \lfloor \frac{L_{p_k}}{2} \rfloor)} (1 - \sum_{b=1}^{2n} \overline{\mathbf{M}}_{P_k}^{SHV}[c, b])).$$

H is the cross-entropy function

Constrain the sum of each line of the SHV matrices to one.

Example:

Logic program: $p \leftarrow q \vee \neg p$

$q \leftarrow \neg p$

(a) A logic program P

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times v_i \xrightarrow{or} \overline{v}_o[0]$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \times v_i \xrightarrow{or} \overline{v}_o[1]$$

(b) Two SHV matrices to represent the logic prgoram P

Model: Learning first-order logic programs

Transferring steps:

- Generate bottom clauses according to the relational database using bottom clause positionalization algorithm[1];
- Regard the bottom clauses as propositional logic programs, and make the first-order features as propositional variables;
- Generate pairs of interpretation transitions.

[1] França, M. V. M., Zaverucha, G., & DAvila Garcez, A. S. (2014). Fast relational learning using bottom clause propositionalization with artificial neural networks. Machine Learning, 94(1), 81–104.

Example

- F = {mother(mom1, daughter1), wife(daughter1, husband1), wife(daughter2, husband2)},
- P = {motherInLaw(mom1, husband1)1}
- N ={motherInLaw(daughter1, husband2)}.

Generated bottom clause set (the depth of the variable is set to 1):

Bottom clause

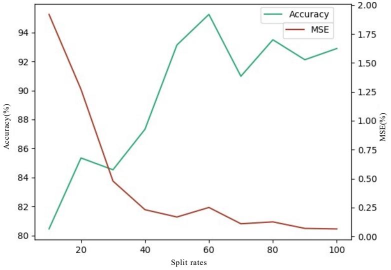
$E_\perp$ = {motherInLaw(A, B) :−mother(A, C), wife(C, B);
∼ motherInLaw(A, B) :−wife(A, C)}.

Pairs of interpretation

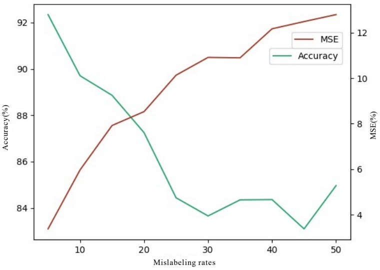
I_1: mother(A, C), wife(C, B)	J_1: motherInLaw(A, B)
I_2: wife(A, C)	J_2:

Results

Test the model on the incomplete and mislabelled datasets:



(a) Incomplete data



(b) Mislabeled data

Figure: Mean accuracy of the logic program and the MSE of the predicted Boolean value with respect to different split rates and mislabeling rates of the fission dataset.

Results

Table 1 Comparison of MSE (%) and accuracy (%) on partial datasets with different split rates

Datasets (Complexity)	Model	The split rates			
		10%	20%	50%	100%
Fission (10, 2^{10})	NN-LFIT	**1.2**, **98.80**	**0.2**, **99.80**	**0.00**,**99.92**	**0.00**,99.87
	D-LFIT	1.91, 80.45	1.27, 85.33	0.17, 93.13	0.06, 92.89
	LFIT	–, 76.85	–,77.03	–,77.15	– ,**100**
	JRip	–,79.14	–,78.05	–,80.04	–, 78.47
Mammalian (10, 2^{10})	NN-LFIT	**1.3**, **96.6**	**0.4** , **94.35**	**0.00**,**99.89**	**0.00**,**99.91**
	D-LFIT	1.73, 71.67	1.21 ,75.9	0.79, 80.09	0.52,82.84
	LFIT	–,76.01	–,76.48	–,76.73	91.56
	JRip	– ,77.84	–,75.44	–,76.41	–, 74.66
Budding (12, 2^{12})	NN-LFIT	ROT	ROT	ROT	ROT
	D-LFIT	1.03, **71.96**	0.43,**71.39**	0.15,**70.5**	0.09,**76.52**
	LFIT	ROT	ROT	ROT	ROT
	JRip	–,67.97	–,68.55	–, 67.91	–,68.32
Arabidopsis (15, 2^{15})	NN-LFIT	ROT	ROT	ROT	ROT
	D-LFIT	0.57, **84.35**	0.51, **86.83**	0.48, **88.56**	0.45, **89.70**
	LFIT	ROT	ROT	ROT	ROT
	JRip	–, 68.84	–, 69.00	–, 68.79	–,68.67
Mutagenesis (1111, 188)	D-LFIT	**77.78**	**94.44**	**88.88**	**83.33**
	JRip	58.37	59.46	65.97	66.45
UW-CSE (601, 1614)	D-LFIT	**75.00**	**78.12**	**78.44**	**79.44**
	JRip	70.18	70.81	73.85	74.35
Alzheimers-amine (1084, 686)	D-LFIT	58.42	**60.29**	**63.24**	**67.65**
	JRip	**58.74**	57.66	57.14	54.81

Table 3 Comparison of the MSE (%) and accuracy (%) on the mislabeled data with different mislabeling rates

Datasets	Model name	The rates of mislabeled data			
		5%	20%	35%	50%
Fission	NN-LFIT	3.25, **96.75**	10.56,**89.43**	15.14,**84.86**	17.74,82.25
	D-LFIT	**2.23**, 92.34	**8.53**, 87.25	**10.89**, 84.34	**12.08**, **84.96**
	LFIT	–,77.25	–,77.16	–,77.38	–,77.32
	JRip	– ,78.91	– ,78.54	–,78.55	– ,78.15
Mammalian	NN-LFIT	4.77, 79.72	16,78.11	20.98,79.01	23.20,74.49
	D-LFIT	**3.45**, **80.00**	**11.53**,**78.82**	**15.74**, **80.29**	**16.35**, **86.27**
	LFIT	–, 76.72	–,76,73	–,76.62	–,77.32
	JRip	–,74.21	–,74.45	– , 74.43	–,74.00
Budding	NN-LFIT	ROT	ROT	ROT	ROT
	D-LFIT	4.9, **76.42**	13.3, **74.28**	16.53,**73.41**	18.21,**74.71**
	LFIT	ROT	ROT	ROT	ROT
	JRip	– ,67.99	–,67.15	–,66.80	–,66.41
Arabidopsis	NN-LFIT	ROT	ROT	ROT	ROT
	D-LFIT	4.8 ,**81.46**	11.83,**76.59**	15.4 ,**70.28**	17.35,64.90
	LFIT	ROT	ROT	ROT	ROT
	JRip	–,68.27	–, 67.34	–,66.94	–,**66.65**
Mutagenesis	D-LFIT	**88.89**	**83.33**	**88.89**	**88.89**
	JRip	66.49	62.23	62.77	62.23
UW-CSE	D-LFIT	**73.49**	**72.67**	**73.29**	**73.29**
	JRip	72.80	67.35	66.04	66.23
Alzheimers-amine	D-LFIT	**63.24**	**63.24**	**60.29**	**58.83**
	JRip	48.35	49.42	49.27	50.87

Comparisons on the incomplete data

Comparisons on the mislabeled data

Table 2 Comparison of the accuracy (%) of rules obtained by CILP++

Model	Datasets		
	Mutagenesis	UW_CSE	Alzheimer-amine
CILP++	77.72	**81.98**	**78.70**
D-LFIT	**83.33**	79.44	67.75

Comparisons on the relational databases

Conclusion

- D-LFIT is an inductive logic programming learner, which can learn propositional logic programs from **mislabeled** data or **incomplete** data.
- Through adopting the BCP algorithm, we can learn **first-order logic programs** from relational databases.
- D-LFIT is a robust, fast leaner, which can **curriculum learning strategy** to learn knowledge from data.
- We will devise more constrains to make the generated logic programs meet destinated formats.
- We will apply the D-LFIT on other dataset such like knowledge graph.

Thanks for your attention!