Neural Logic Reasoning

Shi, Shaoyun, et al. "Neural Logic Reasoning." *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020.

1. Introduction

- 신경망 접근이 좋은 일반화 성능을 보이지만, 여전히 **논리 추리 logical reasoning** 태스크는 잘 다루지 못한다.

- 논리 추리는 “논리 방정식 풀기” 뿐만 아니라 “법조 활동, 개인화 추천 시스템, 의료 활동 보조 시스템” 등을 수행할 때도 사용된다.

  (ex) 추천 시스템

  추리는 사용자와 아이템 간의 복잡한 관계를 모델링 하는 데 도움을 줄 수 있다.
  : User 가 아이템 A 좋아함 & B는 싫어함 → 그러면 유저는 아이템 C를 좋아함

  - 신경망은 드물게rare 나타나는 패턴은 잘 포착하지 못하는 반면, 논리 추리를 이용하면 이러한 것도 잘 포착할 수 있다.

- 머신 러닝 접근이 나타나기 전에는 논리 추리 태스크 시 hard rule-based reasoning 접근을 취했는데, 논리 규칙을 정의하는 것은 어려운 일이었기 때문에 이를 실제로 적용하는 것은 쉽지 않았다.

추론inference은 주어진 전제(Premise)에서 추리reasoning를 통해 결론(Conclusion)을 끌어내는 과정이다. 논증(Argument)은 논리적 추론으로 주장하는 바의 옳고 그름을 증명하는 것이다.

1. Introduction

Background

- **추론**inference은 주어진 전제(Premise)에서 **추리**reasoning를 통해 결론(Conclusion)을 끌어내는 과정이다.
  논증(Argument)은 논리적 추론으로 주장하는 바의 옳고 그름을 증명하는 것이다.

  - **추리**: 추론 활동을 위해 요구되는 심리적 과정. 전제 a,b로부터 c가 도출되는 과정이 심적mental으로 이루어짐

- **명제 논리**

  (∀x)(Hx→Dx): 모든 x에 대하여, x가 사람이면, x는 죽는다
  (∃x)(Hx&Ax): 사람이면서 동물인 x가 존재한다 or 사람이면서 동물인 어떤 x가 있다

➢ 인간이 사용하는 자연 언어로 이루어진 문장 및 논증을 명제 논리만 가지고 표현하는 데 한계가 있음

- **양상modal 논리** (명제 논리의 확장)
  □(∀x)(Hx→Dx): <모든 x에 대하여, x가 사람이면, x는 죽는다>는 필연적이다
  ◇(∃x)(Hx&Ax): <사람이면서 동물인 x가 존재한다>는 가능하다

➢ 양상 논리도 여전히 많은 패러독스와 부딪히고 논리 규칙을 세우는 데 있어서 학자들 간의 합일이 없음

1. Introduction

- DNN과 논리 추리를 통합하기 위해, Logic-Integrated Neural Network (LINN)을 제안한다. 이는 신경망을 기반으로 하여 논리 추론을 수행하는 아키텍처이다.

- LINN은 논리 변수를 표현하기 위해 벡터를 채택하고, 각 기초적인 논리 연산자(and/or/not)은 논리 규칙에 기초하여 신경망 모듈로서 학습된다.

Contributions

1. 논리 추리 활동으로 신경망의 성능을 향상시켜주는 새로운 모델 LINN을 제안한다

2. 실험을 통해 전통적인 신경망과 제안된 모델을 비교한다. LINN은 논리 방정식 해결 태스크에서 기존 모델보다 더 좋은 성능을 보인다.

3. 최신 추천 시스템 모델보다 더 좋은 성능을 보인다.

3. LINN (Logic Integrated Neural Network)

- 논리 변항은 벡터 임베딩으로 표현되고, 논리 연산자(AND/OR/..)은 신경망 모듈로 학습된다.

3.1 신경망 모듈로서의 논리 연산자

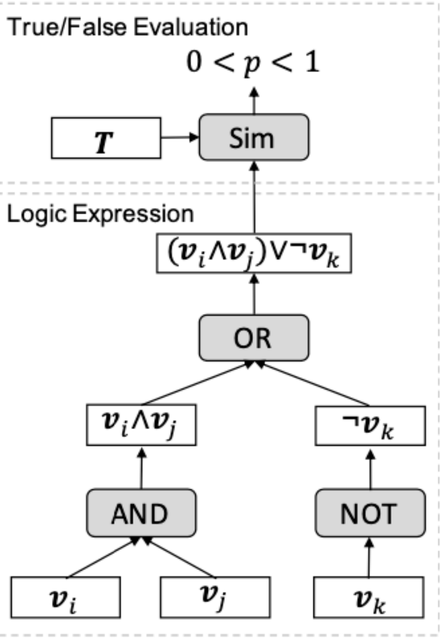
- 명제 논리의 표현은 논리 상항(T or F), 논리 변항(v), 그리고 기초적인 논리 연산자(negation, AND, OR 등)로 구성된다.

A set of logic expression: $E = e_{i}{}_{i=1}^{m}$

Their values(진리치): $Y = y_{i}{}_{i=1}^{m}$ (T,F)

Logic expression을 구성하는 것 (변항들): $V = v_{i}{}_{i=1}^{n}$

ex) $e : (v_i \ \& \ v_j) \ or \ \neg v_k = T$



(a) $(v_i \wedge v_j) \vee \neg v_k$

-T(true)/F도 벡터 표현이다. 이들은 계산된 논리 표현의 진리치와 비교되어 논리 표현이 T인지 F인지 결정할 수 있게 해준다.

- LINN은 다양한 loss 함수로 학습될 수 있다.

  - True/False 예측 문제는 분류 문제로 간주될 수 있음 (크로스엔트로피 채택)

$$L_t = L_{ce} = -\sum_{e_i \in E} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

  - top-n 추천 문제에서는 BPR Loss가 사용될 수 있음

$$L_t = L_{bpr} = -\sum_{e^+} \log\left(sigmoid(p(e^+) - p(e^-))\right)$$

where e_+ are the expressions corresponding to the positive interactions in the dataset, and e^- are the sampled negative expressions.

3.2 신경망 모듈에 대한 논리 규칙화regularization

- 앞에서 우리는 오직 AND, OR, NOT 연산자만을 논리 모듈로 간주했다. 하지만 이 모듈들만 가지고서는 모든 논리 연산이 제대로 수행되지 않을 수 있다.

(ex)

변항 w와 진리치 F의 연언: $w \ \& \ F = F$

이중 부정: $\neg(\neg w) = w$

- 우리는 모델이 위 같은 추가적인 논리 규칙을 학습할 수 있도록 제약을 적용한다.
  - 모듈의 행동behavior을 규칙화하기 위해 논리 규칙자logic regularizer를 정의하고, 이 규칙자가 특정 논리 연산을 수행하도록 만들어준다.

- 3 가지 종류의 규칙자들

Table 1: Logical regularizers and the corresponding logical rules

	Logical Rule	Equation	Logic Regularizer r_i
NOT	Negation	$\neg T = F$	$r_1 = \sum_{w \in W \cup \{T\}} Sim(\text{NOT}(\mathbf{w}), \mathbf{w})$
	Double Negation	$\neg(\neg w) = w$	$r_2 = \sum_{w \in W} 1 - Sim(\text{NOT}(\text{NOT}(\mathbf{w})), \mathbf{w})$
AND	Identity	$w \wedge T = w$	$r_3 = \sum_{w \in W} 1 - Sim(\text{AND}(\mathbf{w}, \mathbf{T}), \mathbf{w})$
	Annihilator	$w \wedge F = F$	$r_4 = \sum_{w \in W} 1 - Sim(\text{AND}(\mathbf{w}, \mathbf{F}), \mathbf{F})$
	Idempotence	$w \wedge w = w$	$r_5 = \sum_{w \in W} 1 - Sim(\text{AND}(\mathbf{w}, \mathbf{w}), \mathbf{w})$
	Complementation	$w \wedge \neg w = F$	$r_6 = \sum_{w \in W} 1 - Sim(\text{AND}(\mathbf{w}, \text{NOT}(\mathbf{w})), \mathbf{F})$
OR	Identity	$w \vee F = w$	$r_7 = \sum_{w \in W} 1 - Sim(\text{OR}(\mathbf{w}, \mathbf{F}), \mathbf{w})$
	Annihilator	$w \vee T = T$	$r_8 = \sum_{w \in W} 1 - Sim(\text{OR}(\mathbf{w}, \mathbf{T}), \mathbf{T})$
	Idempotence	$w \vee w = w$	$r_9 = \sum_{w \in W} 1 - Sim(\text{OR}(\mathbf{w}, \mathbf{w}), \mathbf{w})$
	Complementation	$w \vee \neg w = T$	$r_{10} = \sum_{w \in W} 1 - Sim(\text{OR}(\mathbf{w}, \text{NOT}(\mathbf{w})), \mathbf{T})$

- 위 같은 논리 규칙자는 LINN이 변항에 대한 위 규칙들을 만족하도록 neural module parameters를 학습하게 해준다.
- 논리 규칙자 R_l은 task-specific loss fucntion L_t에 더해진다.
-

$$L_1 = L_t + \lambda_l R_l = L_t + \lambda_l \sum_i r_i$$

3.3 논리 변항에 대한 길이length 규칙화regularization

- 논리 표현 뿐만 아니라 논리 변항의 벡터 길이는 학습 과정 동안에 explode 할 수 있다.
  - 왜냐하면 단순히 벡터 길이를 증가시키는 것은 $L_1 = L_t + \lambda_l R_l = L_t + \lambda_l \sum_i r_i$ 식을 최적화한 것에 대한 자명trivial해만을 산출하기 때문이다.

    ➢ **?**

- 위 이유 때문에, explosion을 방지 하기 위해, **벡터 길이 제약하기**는 stable 성능 제공한다. → l_2 길이 규칙자 R_2가 더해진다.

$$L_2 = L_t + \lambda_l R_l + \lambda_\ell R_\ell = L_t + \lambda_l \sum_i r_i + \lambda_\ell \sum_{w \in W} \|\mathbf{w}\|_F^2$$

w = input variable vectors (expression vectors)

3.3.2 Final Loss

- 오버피팅을 막기 위해 weight λ_θ로 l_2-regularizer를 적용한다. 아래 식에서 Θ는 모델의 모든 parameters라고 가정한다.

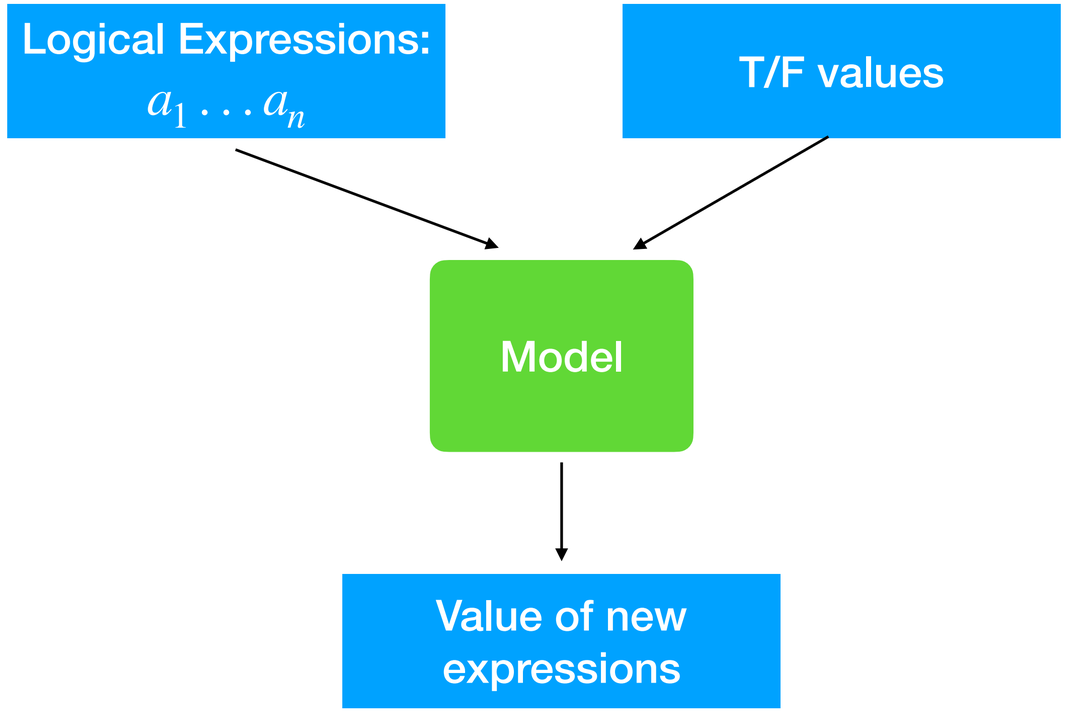
$$\begin{aligned} L &= L_t + \lambda_l R_l + \lambda_\ell R_\ell + \lambda_\Theta R_\Theta \\ &= L_t + \lambda_l \sum_i r_i + \lambda_\ell \sum_{w \in W} \|\mathbf{w}\|_F^2 + \lambda_\Theta \|\Theta\|_F^2 \end{aligned}$$

Logical reg **Length reg**

4. Implementation Details

4.1 Propositional Logical Inference

- **Prototype**



Predict their truth values

- **AND module**

$$\mathrm{AND}(\mathbf{w}_i, \mathbf{w}_j) = \mathbf{H}_{a2} f(\mathbf{H}_{a1}(\mathbf{w}_i \oplus \mathbf{w}_j) + \mathbf{b}_a)$$

$H_{a1} \in R^{d,2d}$

$H_{a2} \in R^{d,d}$

$\oplus = concat$

$f() =$ activation fucntion (ReLU)

- **NOT module**

$$\mathrm{NOT}(\mathbf{w}) = \mathbf{H}_{n2} f(\mathbf{H}_{n1}\mathbf{w} + \mathbf{b}_n)$$

$H_{n1} \in R^{d,d}$

$H_{n2} \in R^{d,d}$

- **Similarity**

$$Sim(\mathbf{w}_i, \mathbf{w}_j) = sigmoid\left(\alpha \frac{\mathbf{w}_i \cdot \mathbf{w}_j}{\|\mathbf{w}_i\|\|\mathbf{w}_j\|}\right) \quad where\ \alpha = 10$$

4. Implementation Details

4.1 Propositional Logical Inference

DNF Dataset

- 임의의 진리치를 가진N 개의 변항을 임의로 만든다.
- M 개의 표현을 “선언or”으로 묶어 임의로 만든다.
  - M 개의 표현은 5 개 이하의 선언지로 구성됨
  - 각 선언지는 5 개 이하의 연언지로 구성됨

$$(\neg v_{80} \wedge v_{56} \wedge v_{71}) \vee (\neg v_{46} \wedge \neg v_{7} \wedge v_{51} \wedge \neg v_{47} \wedge v_{26})$$
$$\vee v_{45} \vee (v_{31} \wedge v_{15} \wedge v_{2} \wedge v_{46}) = T$$
$$(\neg v_{19} \wedge \neg v_{65}) \vee (v_{65} \wedge \neg v_{24} \wedge v_{9} \wedge \neg v_{83})$$
$$\vee(\neg v_{48} \wedge \neg v_{9} \wedge \neg v_{51} \wedge v_{75}) = F$$
$$\neg v_{98} \vee (\neg v_{76} \wedge v_{66} \wedge v_{13}) \vee (v_{97} \wedge v_{89} \wedge v_{45} \wedge v_{83}) = T$$
$$(v_{43} \wedge v_{21} \wedge \neg v_{53}) = F$$

- Cross-Entropy loss 사용
- $n = 10^3$ 개의 변항, $m = 3 * 10^3$ 개의 표현

4. Implementation Details

4.1 Propositional Logical Inference

RESULT

Table 2: Performance on solving logical equations

	$n = 1 \times 10^3, m = 3 \times 10^3$		$n = 1 \times 10^4, m = 3 \times 10^4$	
	Accuracy	RMSE	Accuracy	RMSE
Bi-RNN [32]	0.6493 ± 0.0102	0.4736 ± 0.0032	0.6942 ± 0.0028	0.4492 ± 0.0009
Bi-LSTM [11]	0.5933 ± 0.0107	0.5181 ± 0.0162	0.6847 ± 0.0051	0.4494 ± 0.0020
CNN [19]	0.6380 ± 0.0043	0.5085 ± 0.0158	0.6787 ± 0.0025	0.4557 ± 0.0016
LINN-R_l	*0.8353 ± 0.0043*	*0.3880 ± 0.0069*	*0.9173 ± 0.0042*	*0.2733 ± 0.0065*
LINN	**0.9440 ± 0.0064***	**0.2318 ± 0.0124***	**0.9559 ± 0.0006***	**0.2081 ± 0.0018***

* Significantly better than the best of the other results (italic ones) with $p < 0.05$

논리 제약자 사용 결과

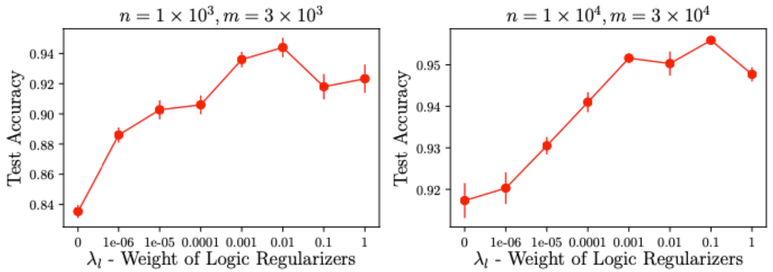


Figure 2: Performance under different choices of the logical regularization weight.

$$L_1 = L_t + \lambda_l R_l = L_t + \lambda_l \sum_i r_i$$

가중치가 높을수록 더 좋은 성능을 보이는 경향이 있음

4. Implementation Details

4.2 Recommender system

Dataset

Table 3: Statistics of the Datasets

Dataset	#User	#Item	#Positive	#Negative
ML-100k	943	1,682	55,375	44,625
Electronics	192,403	63,001	1,356,067	333,121

RESULT

Table 4: Performance on the recommendation task

	ML-100k			Amazon Electronics		
	nDCG@10	Hit@1	time/epoch	nDCG@10	Hit@1	time/epoch
BPRMF [31]	0.3664 ± 0.0017	0.1537 ± 0.0036	4.9s	0.3514 ± 0.0002	0.1951 ± 0.0004	112.1s
SVD++ [21]	0.3675 ± 0.0024	0.1556 ± 0.0044	30.4s	0.3582 ± 0.0004	0.1930 ± 0.0006	469.3s
STAMP [25]	0.3943 ± 0.0016	0.1706 ± 0.0022	8.3s	0.3954 ± 0.0003	0.2215 ± 0.0003	352.7s
GRU4Rec [16]	0.3973 ± 0.0016	0.1745 ± 0.0038	7.1s	0.4029 ± 0.0009	0.2262 ± 0.0009	225.0s
NARM [24]	*0.4022 ± 0.0015*	0.1771 ± 0.0016	9.6s	0.4051 ± 0.0006	0.2292 ± 0.0005	268.8s
LINN-R_l	0.4022 ± 0.0027	*0.1783 ± 0.0043*	20.7s	*0.4152 ± 0.0014*	*0.2396 ± 0.0019*	498.0s
LINN	**0.4064 ± 0.0015***	**0.1850 ± 0.0053***	30.7s	**0.4191 ± 0.0012***	**0.2438 ± 0.0014***	754.9s

* Significantly better than the best of other results (italic ones) with $p < 0.05$