Report for CSE 505 - Phase 3

Xiaofei Sun
Stony Brook University
ID:111753600
xiaofei.sun@stonybrook.edu

***Abstract*—My abstract**

Keywords—Multi-valued Real Logic, Tensor Network, Reasoning.

I. Paper Overview

A. Basic Information

Here is some basic information about the paper I selected:

- Title: Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge[1].
- Author: Luciano Serafini and Artur dAvila Garcez
- From: ArXiv.2016
- Abstract: This paper proposes Logic Tensor Networks (LTN) to integrate **learning** and **reasoning** together based on vector representation. The proposed model LTN represent each object with a vector and then converts each function on multiple objects into a manipulate on their vector representations. Then it also uses a s-norm operator to transform a predicate to a real number in $[0, 1]$, which means the confidence of this predicate. Finally, it defines a lose function for all predicates, which means it transfers the reasoning process into a learning and optimization problem.

B. Definitions

Recall that a first-order language L is composed by three parts:

- $\mathcal{C} = \{c_1, c_2, \dots\}$, the set of constant symbols;
- $\mathcal{F} = \{f_1, f_2, \dots\}$, the set of functional symbols;
- $\mathcal{P} = \{p_1, p_2, \dots\}$ the setof predicate symbols.

C. Groundings

LTN defines a term $\mathcal{G}$, called **grounding**, which contains $\mathcal{G}(c)$, $\mathcal{G}(f)$, $\mathcal{G}(P)$ respect to $c \in \mathcal{C}$, $f \in \mathcal{F}$, $P \in \mathcal{P}$:

Also, LTN define the manipulate of each clause ϕ.

1) Constant: For each constant, LTN allocate a n-dimension vector to it:

$$\mathcal{G}(c) \in \mathbb{R}^n \tag{1}$$

2) Function: For each function f/m on m parameters, LTN define it as a mapping on vector space, which means:

$$\mathcal{G}(f(t_1, t_2, \dots, t_m)) \in \mathbb{R}^{mn} \to \mathbb{R}^n \tag{2}$$

Note that here we specify the parameter of f is t_i instead of c_i because function f can be recursive like $f(f_1(c1, c2), f_2(c1, c3)$.

$$\mathcal{G}(f(t_1, t_2, \dots, t_m)) = \mathcal{G}(f)\mathcal{G}(t_1), \mathcal{G}(t_2), \dots, \mathcal{G}(t_m) \tag{3}$$

More specifically, $\mathcal{G}(f(t_1, t_2, \dots, t_m))$ is fitted by a linear function, which can be implemented with tensor network:

$$\begin{aligned}\mathcal{G}(f(t_1, t_2, \dots, t_m)) =& \mathcal{G}(f)(v_1, v_2, \dots, v_m) \\ =& \mathcal{G}(f)(v) \\ =& M_f v + N_f\end{aligned} \tag{4}$$

where $v = \langle v_1, v_2, \dots, v_m \rangle$, $M_f \in \mathbb{R}^{n \times mn}$, and $N_f \in \mathbb{R}^n$

3) Predicate: Like the grounding of a function,

$$\mathcal{G}(P(t_1, t_2, \dots, t_m)) \in \mathbb{R}^{mn} \to [0, 1] \tag{5}$$

Again, as t_i has been mapped to a vector, then:

$$\mathcal{G}(P(t_1, t_2, \dots, t_m)) = \mathcal{G}(P)\mathcal{G}(t_1), \mathcal{G}(t_2), \dots, \mathcal{G}(t_m) \tag{6}$$

Therefore we transfer a predicate into a series of manipulation on tensor space:

$$\begin{aligned}\mathcal{G}(P(t_1, t_2, \dots, t_m)) =& \mathcal{G}(f)(v_1, v_2, \dots, v_m) \\ =& \mathcal{G}(f)(v) \\ =& \sigma(u_p^T tanh(v^T W_P v + V_P v + B_P))\end{aligned} \tag{7}$$

where $v = \langle v_1, v_2, \dots, v_m \rangle \in \mathbb{R}^{mn}$, $W_P \in \mathbb{R}^{mn \times mn \times k}$, $V_P \in \mathbb{R}^{mn \times k}$, $B_P \in \mathbb{R}^k$, $u_P \in \mathbb{R}^k$, and σ is the sigmoid function.

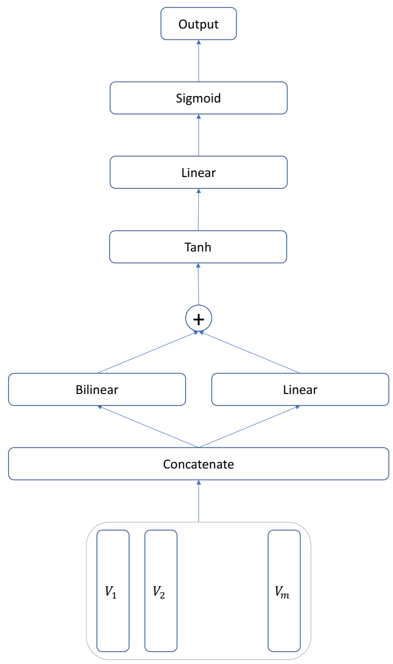


Figure 1: Model for Predicate

4) Clause: In this project, we assume each clause is disjunctive normal form. So it's easy to get that

$$\mathcal{G}(\phi_1, \phi_2, \ldots, \phi_k) = \mu(\mathcal{G}(\phi_1), \mathcal{G}(\phi_2), \ldots, \mathcal{G}(\phi_k)) \quad (8)$$

where μ is the max function.

D. Optimization

So now we get the definition of all components of multi-valued first-order language in tensor networks. The next step is to define a proper lose function.

Saying we know the value of $\phi(x)$ in our dataset, where x is a constant. Intuitively, an optimal grouding should make $\mathcal{G}(\phi(t))$ as close as to $\phi(x)$.

II. Models

A. Logic Tensor Network (LTN)

B. Convolutional Logic Tensor Network (CLTN)

III. Experiments

This section will explore the same example given by the original paper, which is called "friends and smokers" problem. We will firstly how the definition of this example respect to the symbols we use before. Then two experiments will be deducted on knowledge base with only observed facts and knowledge base with extra rules. Finally, we will show the effects of vector length.

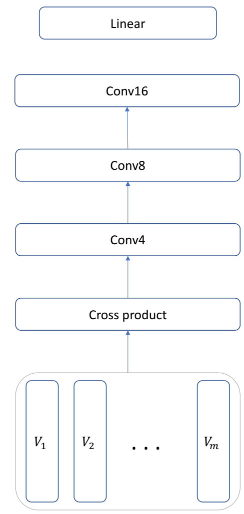


Figure 2: Model for Predicate

A. Friends and Smokers

- Constant $C = C1 \cup C2 = \{a, b, c, d, e, f, g, h\} \cup \{a, b, c, d, e, f, g, h\}$ is all people in two groups $C1$ and $C2$
- Functions: $S/1$, whether a person smokes; $F/2$, whether two persons are friends; $C/1$, whether a person have cancer.
- Predicate: all components of all clauses, including $F/2$, $S/1$ and $C/1$
- Clause: original clauses is defined as the yellow part of Figure.3. We need transfer them into disjunctive form.

The value of "observed facts" we know is shown in Figure.3.

B. Fit Knowledge Base

In this experiment, we only use the oberserved facts, which is $K_{exp1} = K^{SFC}_{a \ldots h} \cup K^{SF}_{i \ldots n}$.

As there is no rules in this part, so we only compare the result of LTN and C-LTN. Figure 4 shows the result of this experiment, where Figure 4a is the probability change with respect to the training epoch, Figure 4b shows the accuracy on the first group and Figure 4c is that of the second group.

From the Figure 4, it's clear that both LTN and C-LTN fit the observed data very well. The reason why the accuracy on first group is not 100% is that there exists a contradictory pair: $S(g)$ and $\neg S(g)$. So it's not possible to make both of them 100% true.

C. Learn From Rule

Firgure 5 shows the result of two models on datasets. Like the result in last subsection, this figure containts three subfigures, including probability changes and best accuracy on two groups.

$$\mathcal{K}^{SFC}_{a\ldots h}$$

$$\begin{array}{r} S(a),\ S(e),\ S(f),\ S(g),\\ \neg S(b),\ \neg S(c),\ \neg S(d),\ \neg S(g),\ \neg S(h),\\ F(a,b),\ F(a,e),\ F(a,f),\ F(a,g),\ F(b,c),\\ F(c,d),\ F(e,f),\ F(g,h),\\ \neg F(a,c),\ \neg F(a,d),\ \neg F(a,h),\ \neg F(b,d),\ \neg F(b,e),\\ \neg F(b,f),\ \neg F(b,g),\ \neg F(b,h),\ \neg F(c,e),\ \neg F(c,f),\\ \neg F(c,g),\ \neg F(c,h),\ \neg F(d,e),\ \neg F(d,f),\ \neg F(d,g),\\ \neg F(d,h),\ \neg F(e,g),\ \neg F(e,h),\ \neg F(f,g),\ \neg F(f,h),\\ C(a),\ C(e),\\ \neg C(b),\ \neg C(c),\ \neg C(d),\ \neg C(f),\ \neg C(g),\ \neg C(h) \end{array}$$

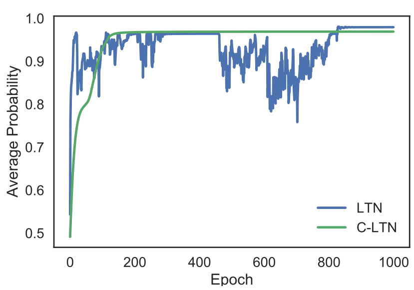
$$\mathcal{K}^{SF}_{i\ldots n}$$

$$\begin{array}{r} S(i),\ S(n),\\ \neg S(j),\ \neg S(k),\\ \neg S(l),\ \neg S(m),\\ F(i,j),\ F(i,m),\\ F(k,l),\ F(m,n),\\ \neg F(i,k),\ \neg F(i,l),\\ \neg F(i,n),\ \neg F(j,k),\\ \neg F(j,l),\ \neg F(j,m),\\ \neg F(j,n),\ \neg F(l,n),\\ \neg F(k,m),\ \neg F(l,m) \end{array}$$

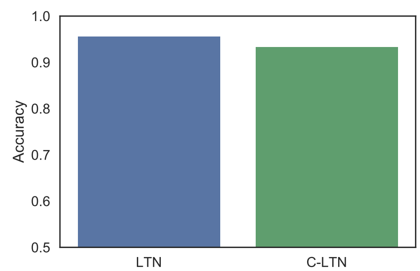
$$\mathcal{K}^{SFC}$$

$$\begin{array}{rr} \forall x \neg F(x,x), & \forall xy(S(x) \wedge F(x,y) \rightarrow S(y)),\\ \forall xy(F(x,y) \rightarrow F(y,x)), & \forall x(S(x) \rightarrow C(x))\\ \forall x \exists y F(x,y), & \end{array}$$

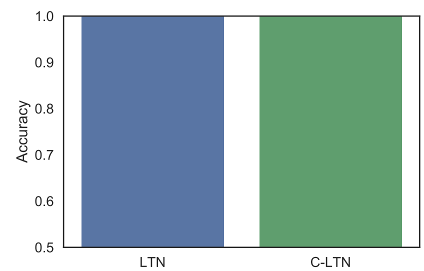
Figure 3: Friends and Smokers



(a) Probability w.r.t. Epoch

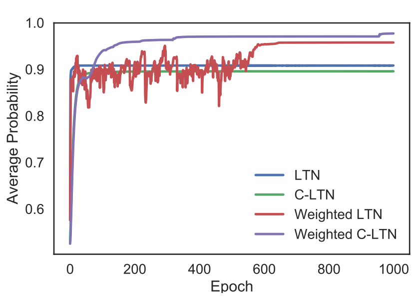


(b) Best Accuracy on Group1

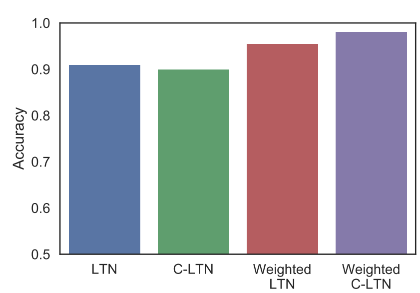


(c) Best Accuracy on Group2

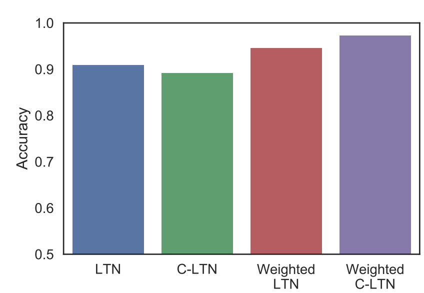
Figure 4: Fitting Observed Facts.



(a) Probability w.r.t. Epoch

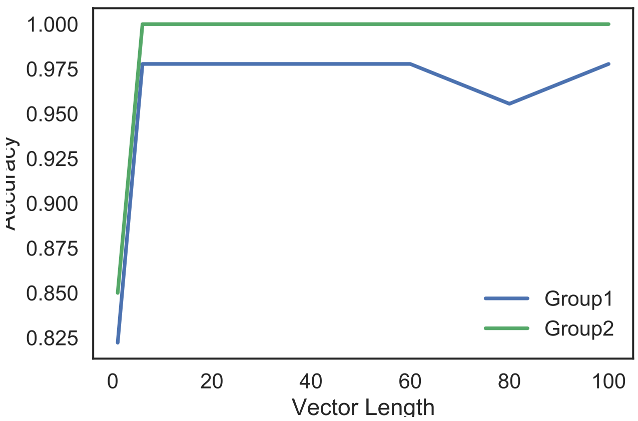


(b) Best Accuracy on Group1

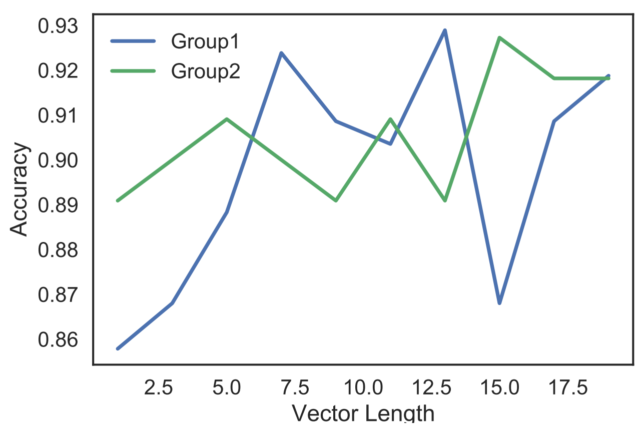


(c) Best Accuracy on Group2

Figure 5: Learning from Observed Facts & Rules.



(a) Fitting Observed Facts



(b) Learning from Rules

Figure 6: Best Accuracy w.r.t. Vector Length.

Some interesting phenomenon can be concluded from the results.

First, by comparing results on weighted and unweighted dataset, we can see that training on weighted dataset gets a better performance. As we said before, this is because the weighted method treat each proposal as one clause, which could avoid the unbalanced training.

Second, C-LTN on weighted datasets shows the best performance. That's because CNN has better fitting ability and more stable in training.

D. Parameter Sensitive

In this part, we want to test the effect of vector length. We enumerate the vector length from 1 to 100 and shows the best accuracy on two groups. We did our experiments on both observed data and weighted dataset with rule. As the performance of C-LNT is better than LNT, we only shows the result of C-LNT, which is shown in Figure 6.

Actually, we want to find the appropriate vector lenght that is enough for fitting the data

IV. Details and Discussion

V. Related Work

Here we try to compare LTN with some other models which try to combine learning and logical reasoning together.

References

[1] L. Serafini and A. d. Garcez, "Logic tensor networks: Deep learning and logical reasoning from data and knowledge," *arXiv preprint arXiv:1606.04422*, 2016.

Model X	Feature of Model X	Feature of LTN
Markov Logic Network	-The level of truth of a formula depends on the number of models that satisfy the formula -Works under the closed world assumption	-The level of truth of a complex formula is -determined by (fuzzy) logical reasoning -Works under open domain
Bayesian Logic	Explicit probabilistic approach	Take the benefits of tensor networks for computational efficiency.
Knowledge Embedding	-Function-free langauges -A special case of LTN -The semantics of the universal and existential quantifiers is based on the closed-world assumption (CWA)	-Provide groundings for functional symbols -A General model -Does not make the CWA -No specific t-norm

Table I: Comparison with Similar Model