Report for CSE 505 - Phase 3

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

***Abstract*—We introcude a model named Logic Tensor Networks (LTN) in this report, which is invented by [1] in their paper "Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge". LTN constructs a logical system with tensor networks, which map the components of a logical system, including constants, functions, predicates and clauses, into the tensor domain. Our contribution include three parts:**

1) **we implement their model;**
2) **we proposed a new structure named Convolutional Logic Tensor Networks (C-LTN), which gets better performance in the "smoker and friends" example;**
3) **we design several experiments to illustrate the performance of logical system implementation with tensor networks.**

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:

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

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

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

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

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)

multilayer perceptron (MLP). The key differencen is that the first layer is not linear layer but bilinear layer.

B. Convolutional Logic Tensor Network (CLTN)

The model in original LTN is based on classical neural networks. However, recent development on deep learning shows that convolutional neural network (CNN) has more powerful ability. That makes us to extend LTN with CNN.

Here we first follow up the implementation of $G(c)$, $G(f)$, and $G(\phi)$. However, we tend to use Convolutional Neural Network to implement $G(p)$.

First we need to construct the input matrix with the $v_1, v_2, \ldots, v_m$. From LTN model, we learned that the bilinear manipulation is critical to get a good performance. Here we construct this bilinear function directly using input vectors of constants. That is, suppose we get m vectors called $x = [v_1, \ldots, v_m]$, then $x \in \mathbb{R}^{k\times m}$, where k is the dimension of constant vectors. Therefore, $x^T x \in \mathbb{R}^{m\times m}$ a squre matrix, which can be treated as a 1-channel image.

The following step is trivial and the model is illustrated in Figure 2

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.

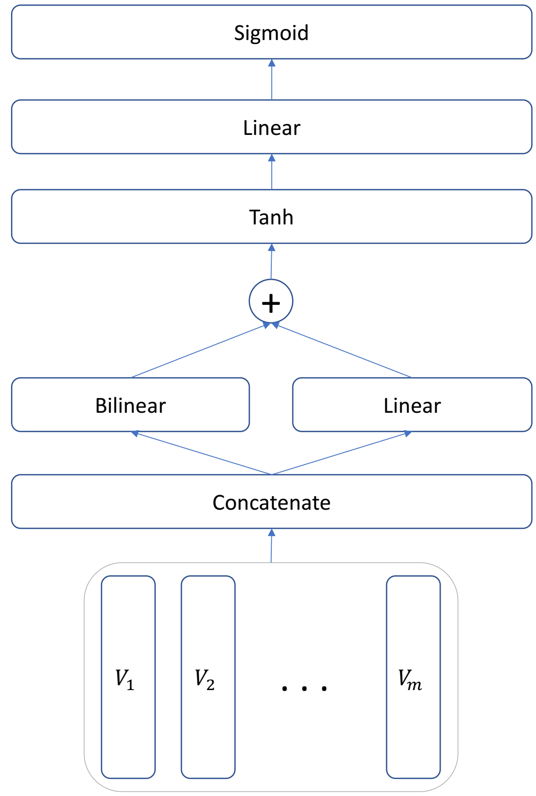


Figure 1: Model for Predicate

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

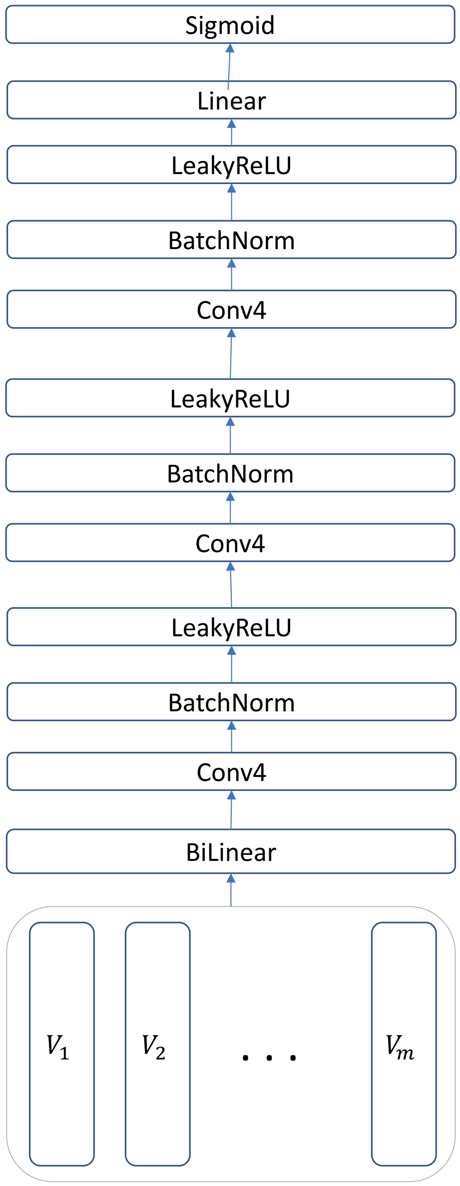


Figure 2: Model for Predicate

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.

Here we also shows the fitting result for each predicate on two groups in Table I and Table II. And it's clear that we have a good fitting to the obseverd data.

Finally, in Table III, we shows the reliability of the rules on this situation. Even though that we didn't use the rules in this experiments, it still got a reasonable probability for each rule.

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.

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

	S	C	F							
			a	b	c	d	e	f	g	h
a	1.00	1.00	1.00	1.00	0.00	0.00	1.00	1.00	1.00	0.00
b	0.00	0.00	1.00	0.11	1.00	0.00	0.00	0.00	0.00	0.00
c	0.00	0.00	0.00	1.00	0.01	1.00	0.00	0.00	0.00	0.00
d	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
e	1.00	1.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
f	1.00	0.00	1.00	0.00	0.00	0.00	1.00	0.01	0.00	0.00
g	0.14	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.08	1.00
h	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00

Table I: Fitting on Group1

	S	C	F					
			i	j	k	l	m	n
i	1.00	0.05	0.74	1.00	0.01	0.01	1.00	0.00
j	0.00	0.02	1.00	0.00	0.00	0.00	0.00	0.00
k	0.00	0.09	0.01	0.00	0.02	0.90	0.01	0.03
l	0.00	0.02	0.01	0.00	0.90	0.01	0.01	0.01
m	0.00	0.11	1.00	0.00	0.01	0.01	0.51	1.00
n	1.00	0.05	0.00	0.00	0.03	0.01	1.00	0.02

Table II: Fitting on Group2

Propositional	Group1	Group2
F(x, x)	0.75	0.666667
F(x, y) F(y, x)	0.984375	0.944444
F(x, y)	1	1
S(x) F(x, y) S(y)	0.953125	0.916667
S(x) C(x)	0.75	0.6666

Table III: Learned Rules

	S	C	F							
			a	b	c	d	e	f	g	h
a	1.00	1.00	1.00	1.00	0.00	0.00	1.00	1.00	1.00	0.00
b	0.00	0.00	1.00	0.05	1.00	0.00	0.00	0.00	0.00	0.00
c	0.00	0.00	0.00	1.00	0.93	1.00	0.00	0.00	0.00	0.00
d	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
e	1.00	1.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
f	1.00	0.00	1.00	0.00	0.00	0.00	1.00	0.01	0.00	0.00
g	0.49	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.10	1.00
h	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00

Table IV: Fitting and Learning on Group1

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

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

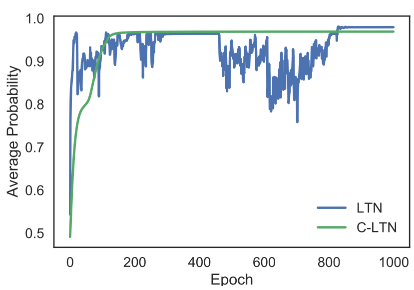
$$\mathcal{K}^{SF}_{i\ldots n}$$

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

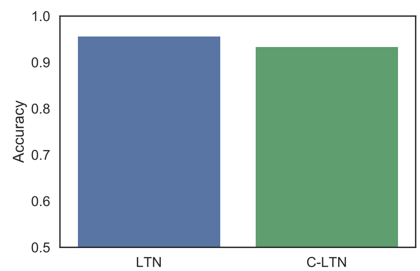
$$\mathcal{K}^{SFC}$$

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

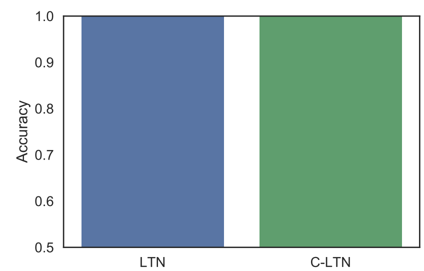
Figure 3: Friends and Smokers



(a) Probability w.r.t. Epoch

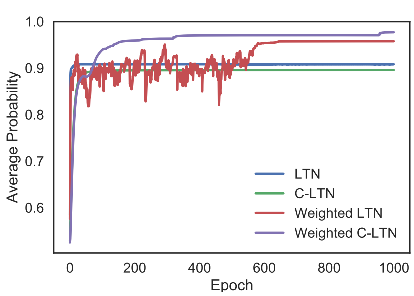


(b) Best Accuracy on Group1

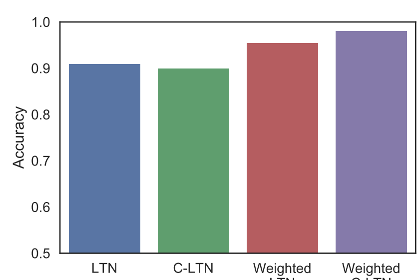


(c) Best Accuracy on Group2

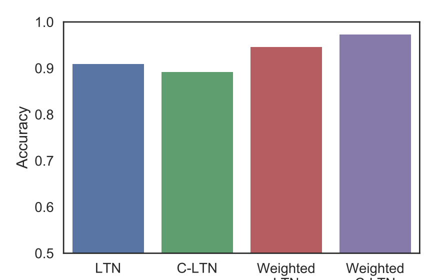
Figure 4: Fitting Observed Facts.



(a) Probability w.r.t. Epoch



(b) Best Accuracy on Group1



(c) Best Accuracy on Group2

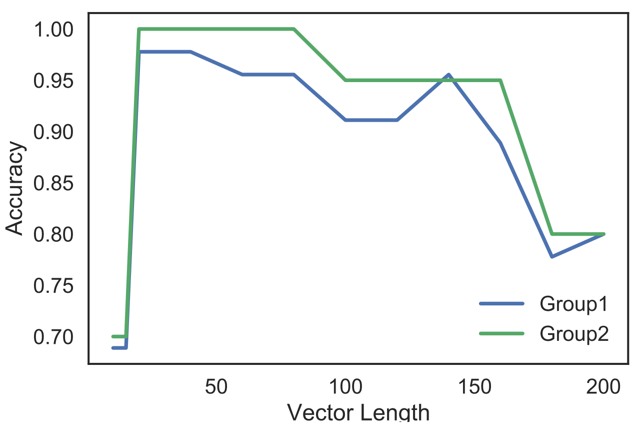
Figure 5: Learning from Observed Facts & Rules.

	S	C	F					
			i	j	k	l	m	n
i	1.00	0.84	0.13	1.00	0.00	0.00	1.00	0.00
j	0.00	0.25	1.00	0.00	0.00	0.00	0.00	0.00
k	0.00	0.76	0.00	0.00	0.13	1.00	0.00	0.03
l	0.00	0.08	0.00	0.00	1.00	0.05	0.00	0.00
m	0.00	0.54	1.00	0.00	0.00	0.00	0.32	1.00
n	1.00	0.04	0.00	0.00	0.03	0.00	1.00	0.05

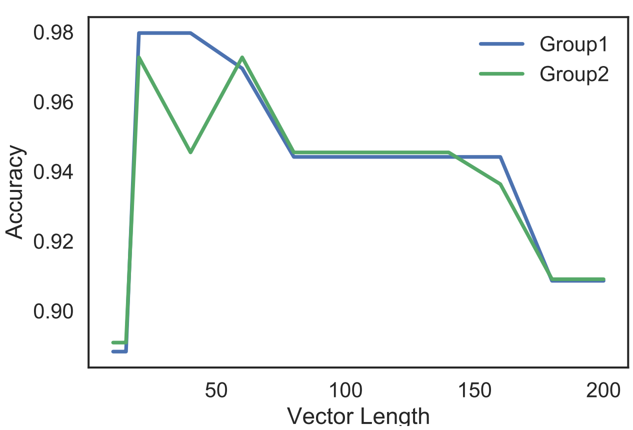
Table V: Fitting and Learning on Group2

Propositional	Group1	Group2
F(x, x)	0.625	0.5
F(x, y) F(y, x)	0.984375	0.916667
F(x, y)	1	1
S(x) F(x, y) S(y)	0.9375	0.916667
S(x) C(x)	0.75	0.666667

Table VI: Learned Rules



(a) Fitting Observed Facts



(b) Learning from Rules

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

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.

Figure 4b shows the result in observed dataset. From this figure, we can conclude that the performance is not always getting better with the increase of vector length. That's may because the under training problem. That is, as the

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 VII: Comparison with Similar Model