Gang Wang*

Automatical Knowledge Representation of Logical Relations by Dynamical Neural Network

DOI 10.1515/jisys-2016-0101

Received July 4, 2016; previously published online September 23, 2016.

Abstract: Currently, most artificial neural networks (ANNs) represent relations, such as back-propagation neural network, in the manner of functional approximation. This kind of ANN is good at representing the numeric relations or ratios between things. However, for representing logical relations, these ANNs have disadvantages because their representation is in the form of ratio. Therefore, to represent logical relations directly, we propose a novel ANN model called probabilistic logical dynamical neural network (PLDNN). Inhibitory links are introduced to connect exciting links rather than neurons so as to inhibit the connected exciting links conditionally to make them represent logical relations correctly. The probabilities are assigned to the weights of links to indicate the belief degree in logical relations under uncertain situations. Moreover, the network structure of PLDNN is less limited in topology than traditional ANNs, and it is dynamically built completely according to the data to make it adaptive. PLDNN uses both the weights of links and the interconnection structure to memorize more information. The model could be applied to represent logical relations as the complement to numeric ANNs.

Keywords: Neural network, knowledge representation, logical relation, probability, dynamic network structure.

2010 Mathematics Subject Classification. 68T05, 68T30.

1 Introduction

With information technology being widely used for many years, things to be solved by computers are becoming more and more complex. Complex things mean not only that the objects to be addressed are complicated and variable, but also that they are in complicated and variable environments. This situation requires programming numerous rules, which make programming manually more and more difficult. The requirement becomes urgent to develop and improve the technology in artificial intelligence (AI) so as to form rules automatically and increase the adaptability of software. Currently, artificial neural network (ANN) becomes a hot research topic again after several up-and-downs. It mimics the biological neural network or the brain, which shows high intelligence and high adaptability. ANN represents and memorizes the relations of objects automatically by using interconnected neurons to exchange messages inside the network. With the development of ANN for many years, considerable ANN models [1, 9, 17, 18] have been proposed, such as the back-propagation neural network [16] and the convolutional neural network [6, 11]. Currently, the ways to fulfill automatic representation and memorization of relations in most ANNs are based on functional approximation, which integrates many neurons' activation functions and numeric weights in connections. In this paper, we refer to them as numeric ANN. In reality, not only numeric relations are used to describe the ratio between things, but also logical relations embodied in *if-then* rules used to describe the sequence of things solved by another AI branch symbolic logic [12, 15]. However, current numeric ANN is not good at representing logical relations.

*Corresponding author: Gang Wang, Department of Computer Science and Technology, Tongji University, Shanghai, China; and Institute of Machine Learning and Systems Biology, Tongji University, Shanghai, China, e-mail: f_lag@buaa.edu.cn

Actually, the biological neural network is good at representing logical relations – even better than representing numeric relations.

Researchers have tried to improve ANN to represent logical relations [2, 3]. This field of “neural-symbolic computation” aims at combining symbolic logic with ANN, the two fields of which are yet developing individually. Through research, they want to apply ANN into more application areas like knowledge representation and reasoning, expert systems, semantic web, and cognitive modeling and robotics. Several approaches have been proposed [3, 4, 7, 8, 13, 19]. A kind of logical neural network composed of neurons of two types is presented [13]. The two types of neurons are defined by two different threshold functions to perform basic logical operations, in which one threshold is 0 and the other is 1. The proposed model represents logical relations based on their combination. However, in reality, completed information cannot be obtained at once, limited by the perceiving capacity. It brings incomplete and uncertain logical relations. The knowledge or relations need to be gradually mastered by learning the instances increasingly and continuously. This neural network model is for fixed numbers of variables stated in this paper, which lacks the probabilistic mechanism to deal with the uncertainty of logical relations, and also lacks the dynamic structures for receiving new perceived things and attributes. The knowledge-based artificial neural network (KBANN) [19] and the connectionist inductive learning and logic programming system [7, 8] based on KBANN set weights and thresholds such that the network behaves as a set of AND/OR neurons. Each rule is mapped from the input layer to the output layer of ANN through one hidden neuron performing a basic logical operation. However, the mapping is inflexible to adjust the neural network structure correspondingly to represent incomplete logical relations in the process of incremental learning under complicated and variable environments. The hidden logical neurons in these ANN models make neurons represent not only things but also the relations between things. The neurons in these models are beyond the boundary of responsibility, as relation representation between things should be done by the links that represent relations between things through connecting neurons. There are some other ANN models proposed to address logical relations in specific domains. For example, a kind of ANN was proposed, specified for recognizing logical relations of two words, phrases, or sentences [4]. However, the specialization pays the universality as the price and limits the application areas of ANN.

In this paper, we propose a novel neural network model called the probabilistic logical neural network (PLDNN), to try to make ANN represent logical relations of things directly. Inhibitory links are defined in addition to exciting links to represent logical relations directly. The links and their weights are adjusted dynamically according to the data, causing PLDNN to have high adaptability to represent logical relations in variable and uncertain environments. We hope that this neural network model could be used as the complement to the current numeric ANN to deal with logical issues and expand the application areas of ANN.

2 Origin of Problems to Represent Logical Relations by ANN

In the way of using neurons connected together to represent logical relations, it is natural that a neuron is used to represent a thing, and the directed link is used to represent a directed relation, like the causality between two things represented by the linked neurons shown in Figure 1. When things happen

Neural network to represent the relation $A \rightarrow B$

E.g. A: Long nose, B: Elephant,

$A \rightarrow B$ means an animal with a long nose is an elephant

Figure 1: Using Components of ANN to Represent Logical Relations.

and are perceived by ANN, neurons representing them are activated by perceiving systems, then these neurons will activate the directed link neurons to make ANN have the ability of reasoning and prediction. For example, in Figure 1, when thing A happens and is perceived by ANN, the neuron representing A is activated. Then, the neuron representing A activates the directed linked neuron representing B to make the ANN know B will happen after A happens. This make ANN have the ability of representing the logical relation $A \rightarrow B$. For conciseness, in the following we use the same identity for the thing and the neuron representing the thing. For example, the identity A not only refers to thing A, but also to the neuron representing A.

Two simple ANNs representing the relation $A, B \rightarrow D$ and $A, C \rightarrow E$ individually are shown in the upper part of Figure 2. The two ANNs can still work individually to represent the relations; however, when they are integrated into a single ANN to represent the two relations in one neural network at the same time, the single ANN cannot work. The obstacle of representing logical relations by ANN is illustrated in the bottom part of Figure 2. When things A and B happen, neurons A and B are activated, then A and B activate the directed linked neurons D and E. It makes the integrated ANN obtain the wrong reasoning result that things D and E will happen after A and B had happened. The integrated ANN represents the wrong logical relation $A, B \rightarrow D, E$ rather than the relation $A, B \rightarrow D$. It is the same situation when things A and C happen. The reason is that neuron A activates all its directed linked neurons. When A and B happen, A should not activate its directed linked neuron E. Similarly, when A and C happen, A should not activate its directed linked neuron D. To make the integrated ANN represent or memorize the right logical relations in one network, there is a need to add some controls to not activate all the directed linked neurons, i.e. inhibiting certain links selectively according to logical relations to make sure that the correction of the ANN represent logical relations and obtain the correct reasoning results.

3 Solutions

When a set of things happen, neurons representing happening things activate the correct sequential neurons. We designed the PLDNN to ensure that the ANN represents the logical relations correctly. The inhibitory mechanism is introduced into the ANN, and the probabilistic mechanism is used to deal with the uncertainty.

Figure 2: Illustrating the Problem of ANN to Represent Logical Relations.

3.1 Logical Components in PLDNN

The basic components in PLDNN are shown in Figure 3. As a subtype of ANN, PLDNN also has two super types: (i) neurons representing the things and (ii) links connecting neurons to represent the relations between things. Unlike the way that current numeric ANNs have one type of link with weights on them to represent the ratio between things, PLDNN has four types of simple links specified for representing logical relations between things. Therefore, from the design aspect, PLDNN has advantages over current numeric ANNs in representing logical relation, because current numeric ANNs have to represent logical relations between things in the essential way of the ratio they work with, while PLDNN is specified for representing logical relations using components representing simple logical relations.

The detailed descriptions of the components of PLDNN are given below.

The neuron is used for representing a thing defined by users; it has three states: resting, positively activated, and negatively activated, indicated by 0, 1 and -1, respectively. In general, the state of the neuron is in the resting state. The word “resting” refers to the term in the biological neuron network. When a thing A happens and is perceived by PLDNN, the state of neuron A becomes positively activated, achieving the goal that PLDNN represents logic A. Similarly, when a thing A does not happen and is perceived by PLDNN, the state of neuron A becomes negatively activated, achieving the goal that PLDNN represents the logic $\neg A$.

There are two types of links in PLDNN – exciting link (EL) and inhibitory link (IL) – in contrast to only one type of link in numeric ANN just containing the ratio between the connected neurons. EL is similar to the link in numeric ANN; the neurons representing happening things excite the post-end neurons through them. IL is different from the link in numeric ANN. Its pre-end connects the neuron, while the post-end connects EL. An IL prevents the neurons representing happening things from exciting their post-end neurons by inhibiting the EL connected by it.

The links in PLDNN also has states. The link has two states: resting and activated, indicated by 0 and 1, respectively. When the link is in the activated state, it can have effects on its post-end. The triggers of these links turning into the activated state are different as follows:

- PEL.state = 1 when its pre-end neuron.state = 1.
- NEL.state = 1 when its pre-end neuron.state = -1.
- PIL.state = 1 when its pre-end neuron.state = 1.
- NIL.state = 1 when its pre-end neuron.state = -1.

Figure 3: Basic Components of PLDNN to Represent Logical Relations.

These triggers are defined according to the requirements of representing logical relations. We take PEL to illustrate. For a PEL, when its pre-end neuron.state=1, the PEL is activated and produce effects on its post-end neuron to represent the logical relation $A \rightarrow B$ shown in Figure 3. When the pre-end neuron.state=0, the PEL is not activated and produces no effects on its post-end neurons.

Multiple simple ELs can be combined together to form composite ELs to fulfill complex excitement. In this way, multiple simple ILs can be combined together to form composite ILs to fulfill complex inhibition. Then, the interaction of the two composite links can represent complex logical relations.

By using the components of PLDNN, the difficulty of putting two logical relations in one ANN in Figure 2 can be solved, as shown in Figure 4. When things A and B happen, neurons A and B are positively activated, and PELs and PILs whose pre-end neurons are A or B are activated next. Then, neurons A and B will excite the directed linked neurons. Neuron A will excite D, E by PEL_{AD} and PEL_{AE} . Neuron B will excite D by the PEL_{BD} . To let PLDNN represent the logical relation $A, B \rightarrow D$, neuron B prevents A from activating E by $PIL_{B,PEL_{AE}}$. The inhibitory mechanism makes PLDNN represent the right logical relations by the interactions between neurons through multiple kinds of links.

3.2 Probabilistic Mechanism in PLDNN

Unfortunately, it is often difficult to obtain exact and complete logical relations. In reality, information is always incomplete, caused by various limits such as the range of perceiving. This is also true in the cognition domain. Complete and certain cognition cannot be obtained at once, and logical relations of things are incomplete and uncertain. The knowledge needs to be mastered gradually by learning the instances increasingly and continuously. To deal with the uncertainty in the process of representing logical relations in PLDNN, probability is introduced into PLDNN, and we put it in the weight of the link to indicate the belief strength of a logical relation between things represented by neurons. It is different from the weights of links in numeric ANN, which represent the ratios between things represented by neurons. The weights in PLDNN are altered continuously to be adaptive for newly coming data.

Principle of weight alteration: The alteration of weights in PLDNN is similar to Hebbian learning [10].

- On the one hand, when a thing happens, the neuron representing the thing turns into the activated state. If the thing represented by the post-end neuron of the neuron's EL actually happens afterwards, and thus the post-end neuron also turns into the activated state, the weight of the EL increases to enhance the strength of activating the post-end neuron, indicating more confidence on the logical relation based on the evidence. At the same time, the weights of the ILs decrease to weaken the inhibitory effect on the EL placed by the ILs in activated state.
- On the other hand, when the pre-end neuron of an EL is in the activated state and the post-end neuron of the EL is not positively activated caused by the actually unhappening things, the weight of the EL decreases to weaken the activation of the post-end neuron, indicating less confidence on the logical relation based on the evidence. At the same time, the weights of the ILs increase to enhance the inhibitory effect on the EL placed by the ILs in activated state.

PLDNN represents multi logical relations in one ANN.

E.g. A: Animal B: Long nose, C: Long neck,

D: Elephant, E: Giraffe

RL: $A, B \rightarrow D$ $A, C \rightarrow E$

Figure 4: Using Components of PLDNN to Represent Logical Relations.

Every kind of link in PLDNN has two counters: NumA and NumB. The weight is the ratio of NumB to NumA:

$$\omega = \text{NumB} / \text{NumA}. \quad (1)$$

The two counters represent different meanings for different links according to the semantics of the links. The ratio of NumB to NumA indicates the belief strength of the logical relation between things represented by neurons used to express the exciting or inhibitory strength of links to make PLDNN reason rightly.

- For PEL(A,B), A, B are the neurons connected by EL. NumA is the number indicating that the pre-end neuron A is in the positively activated state, whereas NumB is the number indicating that the post-end neuron B is in the positively activated state when A is in the positively activated state. The weight indicates the belief strength or probability $P(B=1/A=1)$. The higher the value is, the bigger the exciting strength is, which means that PLDNN reasons more confidently that B will happen given that A happens. The accuracy of reasoning increases continuously by updating the weight with more and more data perceived.
- For NEL(A,B), NumA is the number indicating that the pre-end neuron of the NEL is in the negatively activated state, which is different from the NumA of PEL, whereas NumB is the number indicating that the post-end neuron B of the NEL is in the positively activated state when A is in the negatively activated state. The weight indicates the belief strength or probability $P(B=1/A=-1)$.
- For PIL(A,el), A is a neuron and el is an EL inhibited by PIL. NumA is the number indicating that the pre-end neuron A is in the positively activated state and el is in the activated state, whereas NumB is the number indicating that the weight of el decreases in this condition. The weight indicates the belief strength or probability $P(\text{el.post-end}=-1/A=1, \text{el}=1)$. If el is a type of PEL, the weight further indicates $P(\text{el.post-end}=-1/A=1, \text{el.pre-end}=1)$. If el is a type of NEL, the weight further indicates $P(\text{el.post-end}=-1/A=1, \text{el.pre-end}=-1)$. The higher the value is, the bigger the inhibitory strength is, which means PLDNN reasons more confidently that the thing represented by el.post-end neuron will not happen in that case.
- For NIL(A,el), NumA is the number indicating that the pre-end neuron A is in the negatively activated state and el is in the activated state, whereas NumB is the number indicating that the weight of el decreases in this condition. The weight indicates the belief strength or probability $P(\text{el.post-end}=-1/A=-1, \text{el}=1)$.

Table 1: Weight Alteration for PEL.

Pre-end N	Post-end N	ω_{PEL}	NumA	NumB
1	1	↑	+1	+1
1	-1	↓	+1	

Table 2: Weight Alteration for NEL.

Pre-end N	Post-end N	ω_{NEL}	NumA	NumB
-1	1	↑	+1	+1
-1	-1	↓	+1	

Table 3: Weight Alteration for PIL.

Pre-end N	ω_{EL}	ω_{PIL}	NumA	NumB
1	↑	↓	+1	
1	↓	↑	+1	+1

Table 4: Weight Alteration for NIL.

Pre-end N	ω_{EL}	ω_{NEL}	NumA	NumB
-1	$\uparrow$	$\downarrow$	+1	
-1	$\downarrow$	$\uparrow$	+1	+1

Figure 5: Weight Alteration and Link Creation in PLDNN.

Similarly, the weights of CEL and CIL indicate the belief strength or probability of complex logical relations. The detailed weight alteration of four simple links is shown in Tables 1–4. $\uparrow$ means the weight increases. $\downarrow$ means the weight decreases.

Figure 5A presents a simple example showing how the weights change. When things A, B happen, D happens afterwards. The PLDNN perceives the event, and the network representing the logical relation $A, B \rightarrow D$ is established in the PLDNN for the first time. The values of NumA and NumB of $PEL_{A,D}$ are both 1 to count the things represented by the pre-end neuron A and post-end neuron D (shown in the parentheses of the table in Figure 5A) of $PEL_{A,D}$. $\omega_{PEL_{A,D}}$ is calculated as 1. When things A, B happen again, the PLDNN reasons that D is the result. If D really happens, the counters in $PEL_{A,D}$ are increased by 1 to indicate that A and D have taken place twice. So does $PEL_{B,D}$. If D does not really happen, it is wrong that the PLDNN reasons that D is the result. In this condition, only NumA of $PEL_{A,D}$ is increased by 1; thus, $\omega_{PEL_{A,D}}$, the ratio of NumB to NumA, decreases, weakening the exciting link $PEL_{A,D}$ activating the post-end neuron D. It improves the preciseness of reasoning.

Figure 5B shows the scenario that the PLDNN changes its network structure by creating links and alters the weights of the existing links to be adaptive for representing new logical relations. When things A, C happen, the PLDNN reasons that D is the result according its existing network in Figure 5A. Actually, E is the result of A, C. To memorize the new logical relation $A, C \rightarrow E$, the PLDNN changes dynamically according to the new relation: (i) $PEL_{A,E}$ and $PEL_{B,E}$ are created so as to excite neuron E, and (ii) $PIL_{C,PEL_{A,D}}$ is created to prevent $PEL_{A,D}$ from activating neuron D. The values of NumA and NumB of $PIL_{C,PEL_{A,D}}$ are 1 to count the event that things A and C happen and thing D does not happen afterwards, appearing one time. $\omega_{PIL_{C,PEL_{A,D}}}$ is calculated as 1. $\omega_{PEL_{A,D}}$ decreases because the unhappening thing D causes the NumB of $PEL_{A,D}$ to not increase.

Figure 5C shows the changed network state of the PLDNN after perceiving the event that D happens after things A, B happen a second time. A new IL, $PIL_{B,PEL_{A,E}}$, is created to prevent $PEL_{A,E}$ from activating neuron E.

The weights of the links are updated to indicate the latest belief degrees on the logical relations represented in the PLDNN according to the new perceived event. After learning three times, the PLDNN obtains the knowledge that A is related with D and E, but only A cannot determine whether D or E happens after A happens based on the information of the directed connections and the weights of $PEL_{A,D}$ and $PEL_{A,E}$ shown in Figure 5C. More information like B or C is needed.

The weights in various links are continuously altered according to the data perceived by the PLDNN. They reflect the belief degrees in logical relations. The probabilistic mechanism in PLDNN promotes the fault-tolerant ability. PLDNN can first establish logical relations based on a few data, even only one datum. With more data perceived, including more instances and more attributes, PLDNN changes its network structure and alters weights to adjust for the data. PLDNN automatically and continuously refines itself to improve the preciseness of representing logical relations. PLDNN represents the logical relations of data by the dynamically interconnecting structures of neurons. The mechanisms of creating various links and updating the probabilities provide the adaptation of PLDNN to learn logical relations from incomplete data, and make PLDNN fulfill incremental learning.

4 Process of Activities of PLDNN

Only the neuron and link components are not enough for PLDNN to fulfill the goal of representing logical relations. How to compose them to build the network of PLDNN dynamically according to the data is another essential problem. Just like for buildings or Lego toys, bricks or metals are essential but not enough for a complex construction. The organic composition is necessary to make the construction. Dynamical network building algorithms need designing. It provides PLDNN the ability of learning and adaptability by adjusting the network continuously to memorize new logical relations while perceiving the data unstoppably. The algorithms refer to the process of activities of PLDNN, including the perceiving, associating, and learning of PLDNN.

4.1 Overview of Process of Activities of PLDNN

The general process of the activities of PLDNN is shown in Figure 6. First, PLDNN perceives a datum D1, neurons representing the data are in the activated state, and then PLDNN comes into the associating stage. PLDNN reasons the result of D1 by the interactions of the activated neurons through their ELs and ILs in the network of PLDNN. After the association, PLDNN perceives another datum D2. In semantics, D2 represents the actual results of D1, thus indicating the meaning of $D1 \rightarrow D2$ to PLDNN. In the learning stage, PLDNN alters the network, including connections and weights to represent the logical relations according to the

Figure 6: Overview of the Process of Activities of PLDNN.

consistency of the reasoning set(RS) and actual set(AS). PLDNN repeats the four activities again and again to represent and memorize logical relations.

4.2 Activity of PLDNN: Associating

When a datum D1 is perceived, the neurons representing the things turn into the corresponding states aforementioned in Figure 3 according to the states (happening, unhappening, or unknown) of the things in the datum, then PLDNN comes into the associating stage from the perceiving stage. In the associating stage, PLDNN reasons what are the next happening things in the way that the post-end neurons linked by the ELs of the activated neurons will be added into the reasoning result set RS if the synthetic weights are still above the belief threshold T_{belief} after being inhibited by the related ILs. T_{belief} should be >0.5 , indicating the relation of the probability at least above half is believable, and its highest value is 1. The formal description of the associating activity is as follows. Let $\{o_1, \dots, o_p, \dots, o_n\}$ be the set of activated neurons in PLDNN named ActivatedSet (it means the condition in the logical relation); then, these neurons try to excite the post-end neurons of their ELs and prevent other activated neurons from activating the wrong neurons by ILs based on the network information of PLDNN (including linking structure and weights) to reason the result RS.

In the algorithm to reason RS, ActivatedELSet(o) is the set of ELs in the activated state whose pre-end neuron is o. RAILSet(EL_j) is the set of ILs in the activated state to inhibit EL_j under the condition.

4.3 Activity of PLDNN: Learning

After the associating stage, PLDNN perceives a new datum D2 as the actual result of D1. Then, PLDNN comes into the learning stage to correct its faults, and represents and memorizes the new logical relation by updating the network according to the consistency of the results reasoned by PLDNN with the results that actually happened. Neurons can be divided into four types according to the existence in two sets RS and AS shown in Table 5 according to the consistency. The meaning of Case 1 is as follows: for a neuron not in RS and AS, PLDNN reasons a thing represented by the neuron does not happen, and the thing actually does not happen. It is similar for the other cases.

Table 5: Four Types of Neurons Divided by the Belonging of RS and AS.

Case	RS	AS	Correctness
1	0	0	Right
2	0	1	Wrong
3	1	0	Wrong
4	1	1	Right

Algorithm 1: Associating of PLDNN

```

Initialize  $RS = \emptyset$ 
for each  $o \in \{o_1, \dots, o_p, \dots, o_n\}$  do
  for each  $EL_j \in \text{ActivatedELSet}(o)$  do
    if  $1 - \max_{IL_k \in \text{RAILSet}(EL_j)} (IL_k \omega) > T_{\text{belief}}$  then
      add the post-end neuron of  $EL_j$  into the RS to indicate it is the next
      thing reasoned by PLDNN.
    end if
  end for
end for
end for

```

For the wrong cases, PLDNN should add links to change the network structure to represent and memorize new logical relations to improve the preciseness of reasoning. For the right cases, PLDNN just keeps the current network structure and update the weights aforementioned in Section 3.2.

In detail, the algorithm of adjusting the network structure for Case 1 is as follows. The algorithm for Case 1 enhances the current network structure, which memorizes the relations rightly by weakening the related ELs and strengthening the RAILSet of the ELs. In this way, PLDNN increases the confidence in reasoning what things do not happen given the ActivatedSet representing the condition in logical relation.

The algorithm of adjusting the network structure for Case 2 is as follows. The goal of the algorithm for Case 2 is to add ELs to make PLDNN next time reason the actually happening things that PLDNN does not reason this time.

Algorithm 2: Neural Network Adjusting for Case 1

```

for each  $o_i \in \text{ActivatedSet}$  do
  for each  $el \in \text{ActivatedELSet}(o_i)$  do
    if  $el.\text{post-end} \notin (RS \cup AS)$  then
      weaken  $el$ 
      strengthen  $\text{RAILSet}(el)$ 
    end if
  end for
end for

```

Algorithm 3: Neural Network Adjusting for Case 2

```

for each  $o_i \in (AS - \bigcap_{o_j \in \text{ActivatedSet}} \text{ExcitedSet}(o_j))$  do
  for each  $o_j \in \text{ActivatedSet}$  do
    if  $\text{EL}(o_i, o_j)$  does not exist then
      make  $\text{EL}(o_i, o_j)$ 
    else
      strengthen  $\text{EL}(o_i, o_j)$ 
      weaken  $\text{RAILSet}(\text{EL}(o_i, o_j))$ 
    end if
  end for
  if no new simple EL whose post-end neuron is  $o_j$  is made then
    if  $\text{EL}(\text{ActivatedSet}, o_j)$  does not exist then
      make a composite EL
    else
      strengthen the composite EL
      weaken  $\text{RAILSet}$  of composite EL
    end if
  end if
end for

```

In the algorithm, RAILSet is the related set of ILs in activated state to inhibit the EL. Because the neurons in ActivatedSet have different ELs, making them have ELs with different numbers of neurons in RS, we get their intersection $\bigcap_{o_j \in \text{ActivatedSet}} \text{ExcitedSet}(o_j)$ rather than the union, and AS subtracts the least common set so as to make sure each neuron in ActivatedSet has an EL with the neurons in AS after the adjustment. ExcitedSet is the set reasoned by o_i .

The adjustment of the neural network structure for Case 2 is illustrated in Figure 5A. When the PLDNN perceives that things A, B happen, the neurons representing A and B are positively activated, indicated by $\text{ActivatedSet} = \{A, B\}$. The neurons A, B make interactions by their ELs and ILs in the associating stage. As shown in Figure 5A, A, B have no links initially, so the PLDNN reasons $RS = \emptyset$. Currently, $\text{ExcitedSet}(A) = \emptyset$ and $\text{ExcitedSet}(B) = \emptyset$. When the PLDNN perceives that thing D happens afterward indicated by $AS = \{D\}$, in order to make the network represent the logical relation $A, B \rightarrow D$ and make the PLDNN reason rightly next time, $\text{PEL}_{A,D}$ and $\text{PEL}_{B,D}$ are established.

In the algorithm, if ELs existed, then these ELs are strengthened by increasing the link weights. If no new simple EL connecting the post-end neuron o_j is established to provide more information for representation, then a composite EL will be established to represent a complex logical relation.

The algorithm of adjusting network structure for Case 3 is as follows. The goal of the algorithm for Case 3 is to add ILs to make PLDNN not reason the unhappening things next time.

In the algorithm, PreActivatedELSet is the set of ELs whose post-end neuron is o_j . PActivatedSet is the set of positively activated neurons. NActivatedSet is the set of negatively activated neurons. PreNeuronSet is the set of pre-end neurons of EL.

The algorithm of adjusting network structure for Case 4 is as follows. The algorithm for Case 4 enhances the current network structure, which memorizes the relations rightly by strengthening the related ELs and weakening the RAILSet of the ELs. In this way, PLDNN increases the confidence in reasoning what things happen given the ActivatedSet representing the condition in logical relation.

Here, we describe the algorithms of adjusting network structure in the sequential way of using loops to deal with neurons. Actually, they run in the parallel way.

5 Verification

We carry out experiments and make some analyses using the dataset Zoo [5, 20]. Zoo has animal attributes and animal names. We use it to test whether PLDNN can recall the right animal given the animal attribute conditions, to check whether PLDNN fulfills the goal of automatic logical representation and memorization in neural network at the beginning of the paper. We tailor Zoo into several testing sets for the verification and analyses of the characteristics of PLDNN stated below.

Process of the experiments: PLDNN learns the logical relations between the attributes and the animal names using the strategy of regression test in software engineering. Regression test is a testing strategy to repeat the existing tests to ensure that a change to the system has not broken the existing functionality. In the experiments, the network of PLDNN will change automatically when learning new logical relations between the attributes and a new animal from new Zoo records. When PLDNN learns a new logical relation, we will repeat the previous Zoo records to see whether PLDNN still memorizes the existing logical relations and reasons the right animal. If it does not, PLDNN will learn the previous records again until it gets the right results for all the records.

Algorithm 4: Neural Network Adjusting for Case 3

```

for each  $o_j \in (RS-AS)$  do
  for each  $el \in PreActivatedELSet(o_j)$  do
    for each  $o_i \in PActivatedSet$  do
      if  $o_i \notin PreNeuronSet(el)$  then
        if  $PIL(o_i, l)$  does not exist then
          make  $PIL(o_i, el)$ 
        else
          strengthen  $PIL(o_i, o_j)$ 
          weaken EL
        end if
      end if
    end for
  end for
  for each  $o_i \in NActivatedSet$  do
    if  $PEL(o_i, o_j)$  does not exist then
      if  $NIL(o_i, el)$  does not exist then
        make  $NIL(o_i, el)$ 
      else
        strengthen  $NIL(o_i, el)$ 
        weaken EL
      end if
    end if
  end for
end for

```

```

    end if
  end if
end for
if no new simple IL whose post-end EL is el is made then
  if IL(ActivatedSet, el) does not exist then
    make a composite IL
  else
    strengthen the composite IL
    weaken el
  end if
end if
end for
end for

```

Algorithm 5: Neural Network Adjusting for Case 4

```

for each  $o_j \in (RS \cup AS)$  do
  for  $el \in PreActivatedELSet(o_j)$  do
    strengthen EL(el)
    weaken RAILSet(el)
  end for
end for

```

Three testing sets in incremental sizes are tailored from Zoo and shown in Table 6. In the three experiments, PLDNN automatically represents and memorizes logical relations by forming the corresponding neural network according to the data, and recalls all the animals given the values in the testing set. The figure of the network of PLDNN learning the testing data takes too much space to be shown. Among 20 attributes, the attribute “leg” has six values, “animal type” has seven values, and “size” has three values. The others have two values. The combination of all attribute values makes an enormous problem space. The advantage of PLDNN is that it creates links on demand instead of full connections according to the data.

- The universality and practicability of the PLDNN model in representing logical relations. We use a practical medical data set, Breast Cancer Wisconsin (Original) Data Set, on UCL datasets with 699 instances from Dr. Wolberg’s clinical cases [14]. During the experiment, PLDNN is still workable. It memorizes all the existing logical relations, and reasons the right results on whether the cancer is benign or malignant given the values in the dataset.
- Comparison with the traditional numeric neural network in representing logical relations. We use the Matlab neural network tool to perform experiments with the dataset Zoo; the numeric neural network cannot recall all the right animals given the values in Zoo. It may be caused by the number settings of hidden neurons, but how to determine the number is always a tactic for the numeric neural network. Therefore, PLDNN is better than the numeric neural network in representing logical relations because PLDNN is designed for this purpose.

5.1 Neural Network Structure to Represent Logical Relations

An intuitive demo of the network structure of PLDNN after learning is shown in Figure 7 to explain the characteristics of PLDNN instead of the complex network of PLDNN learning the testing data mentioned above.

Table 6: Results of PLDNN Representing and Memorizing Logical Relations.

Attributes	Animals	Recall degree
10	20	All
15	40	All
20	60	All

Figure 7: Demo: Network Structure of PLDNN to Represent Logical Relations of Animals and Their Attributes.

The PLDNN in Figure 7 can be manually verified on whether PLDNN memorizes the relations based on commonsense. The knowledge graph is represented through the interconnection structure of the neural network. We use colors to distinguish links instead of the legends of links in the paper, for flexibility of programming. The network structure represents logical relations. Green indicates PEL, blue indicates NEL, red indicates PIL, and orange indicates NIL. From Figure 7, we can see that PLDNN creates links according to the data and uses the network structure to represent logical relations.

For example, when PLDNN perceives a data (yellow and black strips), neuron Y representing yellow will excite the following neurons L, T, and G (leopard, tiger, and giraffe) connected by Y's PELs. The neuron BS representing black strips will excite the following neurons T and Z (tiger and zebra) connected by BS's PELs. BS's PILs inhibit Y's PELs to excite neurons L and G, and Y's PIL inhibits BS's PEL to excite neuron Z. After these interactions between neurons, PLDNN reasons that the animal is a tiger given the data (yellow and black strips).

If things have no relations with each other, the neurons representing them have no links between each other. It is different from current numeric ANNs. Most of the numeric ANN models just use information on weights of links whose interconnection is always fixed.

5.2 PLDNN Integration to Accumulate Knowledge

There are requirements of integrating multiple ANNs developed by different organizations into one ANN. It is easy for PLDNN with the high adaptability of a dynamical network. As shown in Figure 7, one PLDNN is specific for recognizing the birds shown in zone A, and another PLDNN is specific for recognizing the mammals shown in zone B. The whole network is the integrated network. For the integration of PLDNN, if things have

no logical relations with each other, the neurons representing them still have no links between each other in the integrated PLDNN. If the knowledge of the two PLDNNs influences each other, make corresponding links when integrating them. The integration of multiple PLDNNs achieves knowledge accumulation.

6 Conclusion

Aiming for representing logical relations by ANN, the paper designs a new ANN, including new neurons, exciting links, and inhibitory links. PLDNN is a dynamical network, creating links and forming the network according to logical relations inside the data. It not only uses the weights on links to memorize information, but also uses the network structure. PLDNN is currently in an incubated state, and the application still has room for improvement. I hope this paper provides the idea to promote the representation ability of logical relations of ANN, and let PLDNN accompany the current numeric ANNs, which are good at image and sound processing, to extend the abilities of ANN for its application into more domains.

Funding: This work was supported by the National Natural Science Foundation of China (grant nos. 61520106006 and 61532008).

Supplements

Figure 7 is used as an intuitive demo to explain the characteristics in Sections 5.1 and 5.2. It is a simple network of PLDNN that memorizes several relations shown in the following. The PLDNN in Figure 7 memorizes them and forms a knowledge graph through the interconnection structure of the neural network. The PLDNN can be manually verified on whether PLDNN memorizes the relations. Reviewers can make verification manually according to the definition of neurons and links in PLDNN, if you are interested [(1)].

- If an animal has hair, then it is a mammal.
- If an animal produces milk, then it is a mammal.
- If a mammal is a predator, then it is a beast.
- If a mammal has a hoof, then it is an ungulate.
- If a mammal is ruminant, then it is an ungulate.
- If an animal has feather and produces eggs, then it is a bird.
- If an animal is airborne, then it is a bird.
- If a beast is yellow and has spots, then it is a leopard.
- If a beast is yellow and has black strips, then it is a tiger.
- If an ungulate has a long neck and long leg, is yellow, and has spots, then it is a giraffe.
- If an ungulate is white and has black strips, then it is a zebra.
- If a bird cannot be airborne, has a long neck, has long legs, and is a mixture of black and white, then it is an ostrich.
- If a bird cannot be airborne, is aquatic, and is a mixture of black and white, then it is a penguin.
- If a bird can be airborne, then it is a swallow.

These relations are usually used as examples in AI-related books and materials like Haykin's *Neural Networks and Learning Machines*.

Bibliography

- [1] Y. Bengio, A. Courville and P. Vincent, Representation learning: a review and new perspectives, *IEEE Trans. Pattern Anal. Mach. Intell.* **35** (2013), 1798–1828.

- [2] T. R. Besold, Same same, but different? Exploring differences in complexity between logics and neural networks, in: *10th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'15)*, Neural-Symbolic.org, Buenos Aires, Argentina, July 2015, 1–6.
- [3] T. R. Besold and K. U. Kuhnberger, Towards integrated neural-symbolic systems for human-level AI: two research programs helping to bridge the gaps, *Biol. Inspired Cognit. Architect.* **14** (2015), 97–110.
- [4] S. R. Bowman, C. Potts and C. D. Manning, *Recursive Neural Networks Can Learn Logical Semantics*, in: *3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, Beijing, July 2015, 12–21, arXiv:1406.1827.
- [5] R. Forsyth, *Pc/beagle user guide*, Pathway Research Ltd., Nottingham, 1987.
- [6] K. Fukushima and S. Miyake, Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position, *Pattern Recognit.* **15** (1992), 455–469.
- [7] A. Garcez, L. Lamb and D. Gabbay, *Neural-symbolic cognitive reasoning, perspectives in neural computing*, Springer, Berlin, 2008.
- [8] A. Garcez, T. Besold, L. Raedt, P. Földiák, P. Hitzler, T. Icard, K. Kuhnberger, L. Lamb, R. Miikkulainen and D. Silver, Neural-symbolic learning and reasoning: contributions and challenges, in: *AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*, Stanford University, CA, 2015.
- [9] S. Haykin, *Neural networks and learning machines*, 3rd ed., Prentice Hall, New York, 2008.
- [10] D. Hebb, *The organization of behavior*, Wiley, New York, 1949.
- [11] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE* **86** (1998), 2278–2324.
- [12] G. F. Luger, *Artificial intelligence: structures and strategies*, 6th ed., Pearson Education, New York, 2008.
- [13] J. Maqāndziuk and B. Macukow, A neural network performing Boolean logic operations, *Opt. Mem. Neural Netw.* **2** (1993), 17–35.
- [14] O. L. Mangasarian and W. H. Wolberg, Cancer diagnosis via linear programming, *SIAM News* **23** (1990), 1–18.
- [15] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems*, 3rd ed., Pearson Education, New York, 2011.
- [16] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back-propagating errors, *Nature* **323** (1986), 533–536.
- [17] J. Schmidhuber, Deep learning in neural networks: an overview, *Neural Netw.* **61** (2015), 85–117.
- [18] Z. Z. Shi, *Intelligence science*, 2nd ed., Tsinghua Press, Beijing, 2013.
- [19] G. G. Towell and J. W. Shavlik, Knowledge-based artificial neural networks, *Artif. Intell.* **70** (1994), 119–165.
- [20] Zoo, *UCI-datasets*, 1990, Available at: <http://archive.ics.uci.edu/ml/datasets/zoo>, Accessed 23 December, 2015.