Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi'an, 2-5 November 2003

THE EQUIVALENCE BETWEEN A CLAUSE SET AND A KIND OF EXTENSION MULTI-LAYER PERCEPTRON

ZHENG PEI, TIAN-MING HUANG, MING QING, KE-YUN QIN

Department of Applied Mathematics, Southwest Jiaotong University, Chengdu, Sichuan 610031, China
E-MAIL: pqyz@263.net

Abstract:

For multi-layer perceptron, its logic function has been discussed by many researchers. This paper simply discusses a kind of propositional calculus with evaluation and an extension multi-layer perceptron model. Based on the propositional calculus, the formulas set can be expressed in the extension multi-layer perceptron. On the contrary, a kind of extension multi-layer perceptron can also be expressed by some formulas set of propositional calculus. This means that, on one hand, semantic deduction of the propositional calculus knowledge base can be implemented by the extension multi-layer perceptron; on the other hand, the logic function of the extension multi-layer perceptron can be showed by formulas set of the propositional calculus.

Keywords:

Multi-layer perceptron; Propositional calculus; Semantic deduction

1 Introduction

Artificial intelligence is not only based on knowledge expression, but also depends on using knowledge. We can get analysis, decision and forecast by using knowledge. In logic system, using knowledge means logic inference or deduction. Now, many logic systems have been proposed as formal models for knowledge expression and reasoning [1-4]. Although propositional calculus is the simplest logic system, it provides with universal method for other logic systems. Reasoning in propositional calculus includes two aspects. The one is called semantic implication, which is connected with interpretation or valuation of propositional algebra of propositional calculus. The other is called syntactic implication, which is decided by axioms and rule of inference of propositional calculus [1]. Generally, we get formulas set from real life that are conflicting in propositional calculus, we can use non-monotonic logic systems to solve the reasoning, and AI has realized that the analysis of the reasoning mechanisms with conflicting formulas is a major task [5-8].

Neural networks are used very popular. As its learning capabilities and parallel computing, neural networks have formed a large research area and are used in many applications [9-13]. For various aims, many kind of neural networks models and algorithms have been proposed [14-17]. We know that neural networks, also known as connectionist models, are systems that try to make use of some of the known or expected organizing principles of the human brain. The big difference between neural network and propositional calculus is that symbolic knowledge representations are used in propositional calculus, but in neural network, the knowledge is expressed in connectionist framework or weights. In [2], the author has proposed an extended version of propositional calculus and has used a connectionist framework, which is called symmetric networks, to represent unknown logic formulas. A connectionist inference engine is then sketched whose knowledge is either compiled from a symbolic representation or learned inductively from training examples. In [18], Wu, et.al has proposed a new kind of reasoning for propositional knowledge, and has showed that every operation in Kleene-Lukasiewicz fuzzy logic can be implemented by some fuzzy neural logic networks. This means Kleene-Lukasiewicz fuzzy logic is a special case of fuzzy neural logic networks.

In [19], an evaluation propositional calculus is proposed, and some properties are discussed. In the paper, we study the equivalence between the logical Formulas set and a kind of extension Multi-layer perceptron, that is, on one hand, we use the extension multi-layer perceptron to represent the formulas of the evaluation propositional calculus and propositional semantic deduction; on the other hand, we study deeply that the extension multi-layer perceptron can be expressed by some formulas set of evaluation propositional calculus. At this point, we think that the extension multi-layer perceptron is equivalence with the formulas set. So, the extension multi-layer perceptron can be identified by logic formulas.

The paper is organized in the following way: firstly, we simply present the evaluation propositional calculus and the extension multi-layer perceptron; secondly, we discusse

0-7803-7865-2/03/$17.00 ©2003 IEEE

the model properties; thirdly, we present an extension neuron and extension multi-layer perceptron, and its logic properties is discussed; finally, we point out that extension multi-layer perceptron can be used to inference and learn an unknown formula set.

2 An Evaluation Propositional calculus and extension multi-layer perceptron

In this section, we simply express the evaluation propositional calculus and the extension multi-layer perceptron.

2.1 The evaluation propositional calculus

Definition 1: The function $\mu: P(X) \to [0,1]$ is called an evaluation function of $P(X)$. For every formula $A \in P(X)$, $\mu(A)$ is called an evaluation of A.

Definition 2: $(P(X), \mu)$ is called an evaluation propositional calculus. $P(X)$ is propositional algebra, and μ is an evaluation function of $P(X)$.

Remark 1: In fact, $P(X)$ is the formula set of Propositional calculus, the function μ can also be comprehended a fuzzy set on $P(X)$. So, every fuzzy set on $P(X)$ is called an evaluation function of $P(X)$, and $\mu(A)$ is the membership degree of A. In the paper, μ of an evaluation propositional calculus is uniquely decided.

Definition 3: Let $\Psi = \{(A_i, \mu(A_i)) | A_i \in P(X)\}$ is an evaluation formula set. $\Delta_\Psi = \{A_i | (A_i, \mu(A_i)) \in \Psi\}$ is called formula set of Ψ, where A_i is a formula of $P(X)$.

Example 1: Let $B = \{p_1 \to p_2, p_1 \to p_3, p_2 \to p_4, p_3 \to \neg p_4, p_1\}$. We can see that B is inconsistent, $p_1 \to p_2$, $p_1 \to p_3$ is a kind of cause that makes B is inconsistent. So, we can define

$\mu'(p_1 \to p_2) = 0.5$, $\mu'(p_1 \to p_3) = 0.5$, $\mu'(p_2 \to p_4) = 1$, $\mu'(p_3 \to \neg p_4) = 1$, $\mu'(p_1) = 1$.

Obviously, the formula that makes B is inconsistent is not unique. According to application, there exist many evaluation functions. In special case, we can define the function μ such that $\forall p \in P(X), \mu(p) = a, 0 < a \le 1$, and there is no difference between $(P(X), \mu)$ and $P(X)$.

All valuations decided by Δ_Ψ are denoted by set Ω_Ψ. If a valuation $v \in \Omega_\Psi$ such that $v(A) = 1$ (A is a formula of Δ_Ψ), then v is called a model of A, denoted $v \models A$. For Δ_Ψ, if $\forall A \in \Delta_\Psi$, $v \models A$, then v is called a model of Δ_Ψ, denoted $v \models \Delta_\Psi$.

Definition 4: The function $V_\Psi : \Omega_\Psi \to R^+$, and

$$V_\Psi(v) = \sum_{v \models A_i} \mu(A_i) \tag{1}$$

is called a truth evaluation function of Ψ, Where $A_i \in \Delta_\Psi$.

Remark 2: Truth evaluation function V_Ψ of Ψ can be used to evaluate valuations. By $V_\Psi(v) = \sum_{v \models A_i} \mu(A_i)$, we can see that if the more A_i is satisfied by $v \models A_i$, the bigger value of $V_\Psi(v)$ is. For two valuation v_1 and v_2, if $V_\Psi(v_1) = V_\Psi(v_2)$, then we think that v_1 and v_2 is the same "Good"; if $V_\Psi(v_1) < V_\Psi(v_2)$, then we think that v_2 is better than v_1.

Definition 5: Let $v \in \Omega_\Psi$, if

$$V_\Psi(v) = \max\{V_\Psi(v') | v' \in \Omega_\Psi\} \tag{2}$$

Then v is called a $V-$model of Ψ. All of the $V-$models of Ψ are denoted by set

$$\Gamma_\Psi = \{v | V_\Psi(v) = \max\{V_\Psi(v') | v' \in \Omega_\Psi\}\} \tag{3}$$

Definition 6: Let $E \subseteq \Delta_\Psi$, E is called a theory of Ψ if and only if E is consistent in Δ_Ψ. The evaluation of a theory E is denoted by
by

$$T_\Psi(E) = \sum_{A_j \in E} \mu(A_j) \tag{4}$$

Definition 7: If $T_\Psi(E) = \max\{T_\Psi(E')\}$, E' is a theory of Ψ, then E is called a $T-$theory of Ψ. All of the $T-$theory of Ψ is denoted by

$$E_\Psi = \{E | T_\Psi(E) = \max\{T_\Psi(E')\}\}$$

Theorem 1[19]: Let E is a $T-$theory of Ψ, E is a maximal consistent set of Δ_Ψ if and only if for every model v of E ($v \models E$) such that $T_\Psi(E) = V_\Psi(v)$.

Corollary 1[19]: v is a $V-$model of Ψ if and only if v is a model of a $T-$theory E of Ψ.

2.2 An extension multi-layer perceptron and its logic property

Definition 8: An extension half-linear function is

$$f(x)=\begin{cases}a, & x\ge a,\\ 0, & x<0,\\ x, & otherwise.\end{cases} \tag{5}$$

where $a\in(0,1]$. In special case, when $a=1$, then $f(x)$ is a half-linear function [18].

There have been the conclusion that if $f(x)$ is a half-linear function, then

$f(x)=\max(\min(x,1),0)=\min(\max(x,0),1)$.

For the extension half-linear function, we have

Theorem 2[19]: Let $f(x)$ is an extension half-linear function, then

$$f(x)=\max(\min(f(a),0)=\min(\max(f(0),a) \tag{6}$$

Theorem 3[19]: Let $f(x)$ is an extension half-linear function, then

$$f(x+c)=\max(\min(f(x)+c,a)-f(-x),0) \tag{7}$$

where, $0<c\le a$ is a constant.

Definition 9: An extension neuron is a processing unit (see Fig.1), and is expressed by

$$y=f(+_{i=1}^{n}(\omega_i\times x_i)+\theta) \tag{8}$$

Where, $x_1,\cdots,x_n\in[0,1]$ are inputs, $\omega_1,\cdots,\omega_n\in[-1,1]$ are weights, θ is threshold, f is an extension half-linear function, and called activation function.

Definition 10: An extension multi-layer perceptron is a neural network $EMLP=(U,W,A,O,NET,ex)$ such that

$U=U_1\cup\cdots\cup U_n$, and $\forall i,j\in\{1,\cdots,n\}$, $U_i\neq\varnothing$, $U_i\cap U_j=\varnothing$, $i\neq j$, U_1 is called the input layer, U_n is called the output layer, $U_2,\cdots,U_{n-1}$ are called hidden layer.

The network structure is given by the function W: $U\times U\to R$, There exists only connections between consecutive layers, that is, $u\in U_i\wedge W(u,s)\neq\varnothing\Rightarrow s\in U_{i+1}$, for $u^1\in U_1$, $u^2\in U_2$, $W(u^1,u^2)\in\{-1,0,1\}$, In fact, W defines weights.

$\forall u\in U_j, j\in\{2,\cdots,n-1\}$,

$$a_u=A_u(NET_u)=f(NET_u) \tag{9}$$

$\forall u\in U_1$ or U_n,

$$a_u=A_u(NET_u)=NET_u \tag{10}$$

A_u is called activation function, f is an extension half-linear function.

$\forall u\in U_k, k\in\{1,\cdots,n\}$, the output is $u_{out}=O_u(a_u)=a_u$. O_u is called output function.

$\forall u\in U_k$, the input is

$$NET_u=+_{s\in U_{k-1}}W(s,u)\times u_{out}+\theta_u \tag{11}$$

θ_u is the threshold. Specially, $\forall u\in U_1$,

$NET_u=ex(u)$

$ex:U_1\to\{0,1\}$

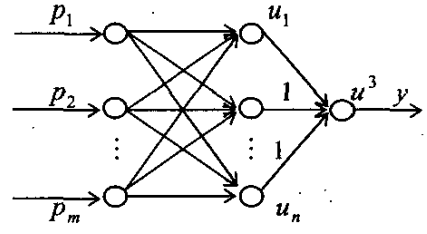

Fig.1. An Extension multi-layer perceptron based on ψ

If we have an evaluation formula set Ψ , and assume $U_1=\{p_1,p_2,\cdots,p_m\}$ is the atomic formula set of Δ_Ψ. According to the definition 10, we can construct an extension multi-layer perceptron based on Ψ (see Fig.1). There are three layers and only one output in Fig.1.

$U=U_1\cup\Delta_\Psi\cup\{u^3\}$.

$W:U\times U\to R$ such that $\forall p_i\in U_1, \forall u_j\in\Delta_\Psi$,

if $p_i\in u_j$, then $W(p_i,u_j)=1$; if $\neg p_i\in u_j$, then $W(p_i,u_j)=-1$; otherwise, $W(p_i,u_j)=0$. $W(u_j,u^3)=1$

$a_{p_i}=A_{p_i}(ex(p_i))=ex(p_i)$, $a_{u_j}=A_{u_j}(NET_{u_j})=f(NET_{u_j})$, $a_{u^3}=A_{u^3}(NET_{u^3})=NET_{u^3}$. A_u is called activation function, f is an extension half-linear function.

$$f_j(x)=\begin{cases}a_j, & x\ge a_j,\\ 0, & x<0,\\ x, & otherwise.\end{cases} \tag{12}$$

and a_j is evaluation of formula u_j.

$$NET_{u_j} = +_{p_i \in U_1} W(p_i, u_j) \times a_{p_i} + \theta_{u_j} \quad (13)$$

$\theta_{u_j} = n_j$ is the threshold, n_j is the number of $W(p_i, u_j)$=-1.

$$NET_{u^3} = \sum_{u_j \in \Delta_\Psi} W(u_j, u^3) \times a_{u_j}.$$

$$ex: U_1 \to \{0,1\}$$

Proposition 1: In Fig.1, if the output $y \neq 0$, then the input is a model of some formulas of Δ_Ψ.

Proof: By $y \neq 0$, we can get

$$NET_{u^3} = \sum_{u_j \in \Delta_\Psi} W(u_j, u^3) \times a_{u_j} \neq 0$$

$\forall u_j \in \Delta_\Psi$, $W(u_j, u^3) = 1$, so, there exists at least one $a_{u_j} \neq 0$, *i.e.*, $\exists u_j \in \Delta_\Psi$, and for inputs of Fig.1, x is such that $f_j(x) = a_j$. So, x is a model of u_j.

Proposition 2: In Fig.1, if the input is a V – model v of Ψ, then the output $y = V_\Psi(v)$.

Proof: By (13), we know that if the input of Fig.1 is a V – mode v of Ψ, then $NET_{u_j} \geq 1$. So, in Fig.1, $y = V_\Psi(v)$.

Fig.1 and proposition 1 and 2 show that an evaluation formula set Ψ can be expressed in an extension multi-layer perceptron, and the extension multi-layer perceptron is very special, that is,

a) $U = U_1 \cup \Delta_\Psi \cup \{u^3\}$ are atomic formula set, clause set and output respectively;

b) the weights are 1, 0 or -1, the extension half-linear function f is decided by the evaluation function of Δ_Ψ;

c) the inputs v are in $\{0,1\}$, the output is $y = V_\Psi(v)$.

3 A clause set decided by extension multi-layer perceptron

Conversely, if or not an extension multi-layer perceptron can be expressed by a logical formula set, that is, in logic view, the extension multi-layer perceptron has the same logic function with a logical formula set. In the section, we only study an extension three-layer perceptron and how to get a logical formula set from being given the extension multi-layer perceptron.

3.1 Adjust an extension three-layer perceptron

Generally, it is difficult to get the formula set of an extension multi-layer perceptron. For an extension three-layer perceptron, we can adjust the extension three-layer perceptron (Fig.2), and get the formula set.

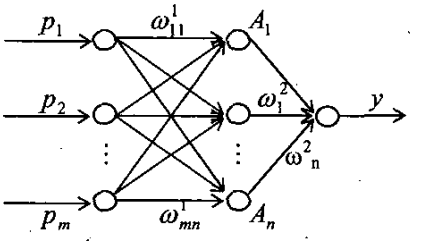

Fig.2. An Extension three-layer perceptron

In fig.2, let $\omega^1_{ij} \in \{-1,0,1\}$, $i = 1, \cdots, m$, $j = 1, \cdots, n$, $\omega^2_n > 0$, and the active function of the neuron A_j is (12). We adjust the extension three-layer perceptron as follows, the active function of the neuron A_j is adjusted as

$$f_j(x) = \begin{cases} \omega^2_j a_j, & x \geq a_j, \\ 0, & x < 0, \\ \omega^2_j x, & otherwise. \end{cases} \quad (14)$$

$$\forall j, \omega^2_j = 1.$$

3.2 A clause set in the neural network

By above adjustment, we can get a logical formula set,

A) every neuron of the input layer is a atom proposition, denoted as $p_1, \cdots, p_m$.

B) every neuron of the hidden layer is a clause, that is,

$A_j = \omega^1_{1j} p_1 \vee \cdots \vee \omega^1_{mj} p_m$, and if $\omega^1_{ij} = 1$, then $\omega^1_{ij} p_i = p_i$; if $\omega^1_{ij} = 0$, then p_i is not contained in A_j, if $\omega^1_{ij} = -1$, then $\omega^1_{ij} p_i = \neg p_i$.

C) if the active function of neuron A_j is an extension half-linear functions, then the evaluation function is $\mu_{A_j} = f_j$.

For the clause set $\Psi = \{A_1, \cdots, A_n\}$, the true evaluation function of Ψ can be gotten from Fig.2, that

is,

$$V_{\Psi}(v)=\sum_{\forall v}\mu_{A_j}(A_j) \tag{15}$$

here, v is inputs of Fig.2..

Generally, the active function of neuron has threshold, that is, the active function is

$$f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j) \tag{16}$$

and θ_j is a threshold. For this case, the active function can be adjusted by $f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j)=f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+(\theta_j-l_j)+l_j)$. Here, l_j is the number of $\omega_{ij}^1=-1$. If adding a neuron θ_j in the input layer of Fig.2, the input of θ_j is always 1 and the weight between θ_j and A_j is θ_j-l_j, then the new network is similar to Fig.2, but its output is not the true evaluation function of Ψ. According to theorem 3, we have

$$f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j)=\max(\min(f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+l_j)+(\theta_j-l_j),\omega_j^2 a_j)-f_j(-(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+l_j)),0) \tag{17}$$

Theorem 4: In Fig.2, if the output of the neuron A_j is $\omega_j^2 a_j$, and $\theta_j-l_j=0$, then the inputs ($\in\{0,1\}$) of Fig.2 is a model of the clause $A_j=\omega_{1j}^1 p_1\vee\cdots\vee\omega_{mj}^1 p_m$.

Proof: By (14),

$f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j)=\omega_j^2 a_j$, then $\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j\ge a_j$. By $\theta_j-l_j=0$, so,

$$\sum_{i=1}^{m-l_j}\omega_{ij}^1 p_i+\sum_{i'=m-l_j+1}^{m}(\omega_{i'j}^1 p_{i'}+\theta_j)\ge a_j,$$

here, $\omega_{i'j}^1=-1$ and $\omega_{ij}^1=1$ or 0, for $a_j\in(0,1]$, we get

$$\sum_{i=1}^{m-l_j}\omega_{ij}^1 p_i\ge 0,\ \sum_{i'=m-l_j+1}^{m}(\omega_{i'j}^1 p_{i'}+\theta_j)\ge 0.$$

So, there exists at lest one term such that $\omega_{ij}^1=1$ and $p_i=1$ or $\omega_{i'j}^1=-1$ and $p_{i'}=0$. This means that the clause A_j is true, that is, input $(p_1,\cdots,p_m)$ is a model of A_j.

Corollary 2: In Fig.2, if $\theta_j-l_j=0$ and the output of Fig.2 is $y=\sum_{j=1}^{n}\omega_j^2 a_j$, then the input $(p_1,\cdots,p_m)$ is a $V-$model of the clause set $\Psi=\{A_1,\cdots,A_n\}$.

Proof: By $y=\sum_{j=1}^{n}\omega_j^2 a_j$, we know that $\forall A_j$, by theorem 4, the input is a model of every A_j, that is, the input $(p_1,\cdots,p_m)$ is a $V-$model of the clause set Ψ.

3.3 Solving $V-$model by programming

According to theorem 4 and corollary 2, the $V-$models of the clause set Ψ can be gotten by solving the following $0-1$ programming problem:

$$\max\sum_{j=1}^{n}f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j)$$
$$s.t.\begin{cases}\omega_{11}^1 p_1+\cdots+\omega_{m1}^1 p_m+\theta_1\ge a_1\\ \vdots\\ \omega_{1n}^1 p_1+\cdots+\omega_{mn}^1 p_m+\theta_n\ge a_n\end{cases} \tag{18}$$
$$p_i\in\{0,1\},\omega_{ij}^1\in\{-1,0,1\},i=1,\cdots,m,j=1,\cdots,n$$

Theorem 5: If $(p_1,\cdots,p_m)$ is an optimal solution of (18), then $(p_1,\cdots,p_m)$ is a $V-$model of the clause set Ψ, and $V_{\Psi}(v)=\sum_{j=1}^{n}\omega_j^2 a_j$.

Proof: If $(p_1,\cdots,p_m)$ is an optimal solution of (18), according to the restricted conditions of (18), we know that $(p_1,\cdots,p_m)$ is model of every A_j, so, conclusion is hold.

If the clause set Ψ is unsatisfiable, then the programming (18) has no optimal solutions. So, we modify (18) as follows:

$$\min(E=\sum_{j=1}^{n}(\omega_j^2 a_j-f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j)))$$
$$s.t.\begin{cases}\omega_{11}^1 p_1+\cdots+\omega_{m1}^1 p_m+\theta_1\ge 0\\ \vdots\\ \omega_{1n}^1 p_1+\cdots+\omega_{mn}^1 p_m+\theta_n\ge 0\end{cases} \tag{19}$$
$$p_i\in\{0,1\},\omega_{ij}^1\in\{-1,0,1\},i=1,\cdots,m,j=1,\cdots,n$$

Corollary 3: If the optimal value of (19) is $E=0$, then the optimal solutions of (19) is a $V-$model of the clause set Ψ, and $V_{\Psi}(v)=\sum_{j=1}^{n}\omega_j^2 a_j$.

Proof: By $E=0$, we can get

$\sum_{j=1}^{n}(\omega_j^2 a_j-f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j))=0$, by (14), we get $\omega_j^2 a_j-f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j)\ge 0$, so, we have $f_j(\omega_{1j}^1 p_1+\cdots+\omega_{mj}^1 p_m+\theta_j)=\omega_j^2 a_j$, by

theorem 5, conclusion is hold.

Corollary 4: If the optimal value of (19) is $E \neq 0$, then optimal solutions of (19) is a $V-$model of the clause set,

$$\Delta = \left\{ A_j \middle| A_j = \omega_{1j}^1 p_1 \vee \cdots \vee \omega_{mj}^1 p_m,\ \omega_{1j}^1 p_1 + \cdots + \omega_{mj}^1 p_m + \theta_j \geq a_n \right\} \text{ and } V_\Delta(v) = \sum_{j=1}^{n} \omega_j^2 a_j - E' .$$

Where $E' = \sum_{j' \in S} \omega_{j'}^2 a_{j'}$, $S = \left\{ j \middle| f_j(\omega_{1j}^1 p_1 + \cdots + \theta_j) = 0 \right\}$.

4 Conclusion

In the paper, we mainly discuss an extended version of propositional calculus and some properties of it. Then, we present an extension multi-layer perceptron model that formulas of the extended calculus can be expressed in it. On using the network to inference and learn, we only give some simple conclusions. We know that the neural network is a useful tool for dealing with data information. So we think that its learning and parallel computing can help us to solve some logic problems.

The authors would like to thank the National Natural Science Foundation of People's Republic of China with Grant No. 60074014.

References

[1] Donald W. Barnes, and John M. Mack, "An Algebraic Introduction to Mathematical Logic", Springer-Verlag New York Inc, 1975

[2] Gadi Pinkas, "Reasoning, Nonmonotonicity and Learning in Connectionist Networks that Capture Propositonal Knowledge", Artificial Intelligence, 77: 203-247, 1995

[3] M.A.Cardenas Viedma, R. Marin Morales and I. Navarrete Sanchez, "Fuzzy Temporal Constraint Logic: A Valid resolution principle", Fuzzy Sets and Systems, 117:231-250, 2001

[4] Paola Forcheri, Paopo Gentilini and Maria Teresa Molfino, "Informational Logic as A Tool for Automated reasoning", Journal of Automated Reasoning, 20:167-190, 1998

[5] Y.Boufkhad, "Algorithms for Propositional KB Approximation", In Proceedings of the Fourteenth National Conference on Artificial Intelligence, pp 280-285, 1998

[6] Madiso. WI. R.Khardon, D. Roth, "Reasoning with Models", Artificial Intelligence, 87:187-213, 1996

[7] Zheng Pei, Jun Liu and Yang Xu, "A kind of Resolution by Deleting in Fuzzy Neural Network", In Proceeding East West Fuzzy Colloquium and the Eighth of Zittau Fuzzy Colloquium, Zittau. Germany pp 238-244, 2000

[8] Zheng Pei, Haiming Li and Yang Xu, "A Kind of Resolution Based on NN", The Eighth of International Conference on Neural Information processing, Shanghai, China, Fudan University Press, pp 941-947, 2001

[9] Kuhu Pal and Nikhil R. Pal, "A Neuro-Fuzzy System for Inferencing", International Journal of Intelligent Systems, 14:1155-1182, 1999

[10] J.D.Horton, Bruce Spencer, "Clause Trees: A Tool for Understanding and Implementing Resolution in Automated Reasoning", Artificial Intelligence, 92:25-89, 1997

[11] Erica Melis and Jorg Siekmann, "Knowledge -based Proof Planning", Artificial Intelligence, 115:65-105, 1999

[12] M. L. Vaughn, "Derivation of The Multi-layer Perceptron Weight Constraints for Direct Network Interpretation and Knowledge Discovery", Neural Networks, 12:1259-1271, 1999

[13] Ilona Jagieska, Chris Matthews and Tim Whitfort., "An Investigation into The Application of Neural Networks, Fuzzy Logic, Genetic Algorithms, and Rough Sets to Automated Knowledge Acquisition for Classification Problems", Neurocomputing, 24:37-54, 1999

[14] Mohamed Tajine and David Elizondo, "The Recursive Deterministic Perceptron Neural Network", Neural Networks, 12:1571-1588, 1998

[15] James Dunyak and Donald and Wunsch, "Fuzzy Number Neural Networks", Fuzzy Sets and Systems, 108:49-58, 1999

[16] Salvatore Cavalieri and Orazio Mirabella, "A Novel Learning Algorithm Which Improves The Partail Fault Tolerance of Multilayer Neural Networks", Neural Networks, 12:91-106, 1999

[17] James J. Buckley and Yoichi Hayashi, "Fuzzy Neural networks: A survey", Fuzzy Sets and Systems, 66:1-13, 1994

[18] Wangming Wu and Hoon-Heng Teh, "Reasoning with Propositional Knowledge based on Fuzzy Neural Logic", International Journal of Intelligent Systems, 11:251-265, 1996

[19] Zheng Pei, "Doctoral Thesis: Study on Automated Reasoning Theory and Method Based on Neural Network", Southwest Jiaotong University, China, 2