Decision-Support Neural Networks and Its Applications in Financial Forecast

XingHu ZHANG, Francis WONG, PeiZhuang WANG
Institute of Systems Science, National University of Singapore, Kent Ridge, Singapore 0511
Fax: (65)778-2571, E-mail: ISSZXH@NUSVM.Bitnet

Abstract

In this paper we will establish a new kind of neural networks, Decision-Support Neural Networks (DSNN) based on truth value flow inference(TVFI)[2][5], approximate reasoning based on similarity theory[6], neural networks knowledge[7][8], and possibility theory[9]. As a succeeding work of "stock selection strategy using fuzzy neural networks"[1], in this paper we also apply this neural networks in financial forecast. As a decision making tool, this neural networks actually can be used in many decision-making fields.

1. Building the Decision-Support Neural Networks(DSNN):

First we give out the diagram of this neural networks (Fig.-1), then we further illustrate its meaning in detail. For convenience, we call this neural networks the Decision-Support Neural Networks (DSNN). And in order to write clearly, we give every layer a name. From input layer to output layer, they are "Factor Layer", "Predicate Layer", "Rule Layer", "Evaluation Layer" and "Decision Layer" respectively as shown in the Fig.-1.

In the Decision-Support Neural Networks (DSNN), the **factor (input) layer** represents the 12 accounting items that is used in the evaluation of stocks. The symbols used in the factor layer are INITIALs of the following 12 accounting items respectively.

1. Issued Capital (IC);	2. Price (P);
3. Market Capitalization (MC);	4. Earnings Per Share (EPS);
5. Price Earnings Ratio (PER);	6. Dividend Yield (EY)
7. Dividend Yield (DY);	8. Net Asset Backing (NAB);
9. Debt-Equity Ratio (DER);	10. Dividend Times Covered (DTC);
11. Return on Shareholders Equity (RSE);	12. Turnover Growth Rate (TGR).

The **predicate layer** consists of 60 fuzzy predicates which are distributed in 12 different factor-spaces. For example, for factor PER, its factor-space is the Real line (Fig.-2) and the five fuzzy predicates distributed in this factor-space have the representations shown in Fig.-3.

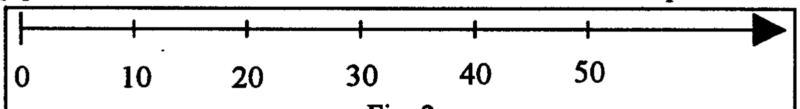


Fig. 2

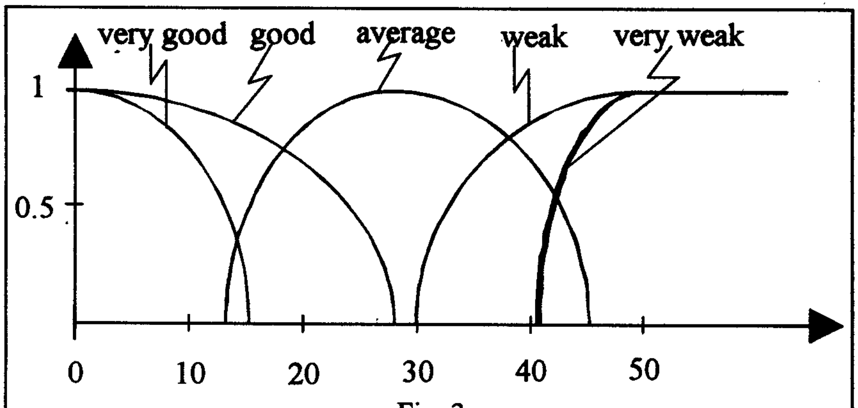


Fig. 3

Here we use "very high(VH)", "high(H)", "average(A)", "low(L)", "very low(VL)" to express these five fuzzy predicates.

The **rule layer** is the third layer, in this layer each node represents a rule, and we use a neural networks for logic operations to implement the logic operations the rules involved. This method is described in detail in the later part.

The **evaluation layer** is the fourth layer, in this paper we use five fuzzy quantifiers as evaluation values, they are "very good(VG)", "good(G)", "average(A)", "bad(B)", "very bad(VB)".

The **decision layer** is the output layer, in this layer three decision strategies "buy", "hold", and "sell" form the strategy set.

Now we define all the weights in the DSNN.

The weights between the factor layer and the predicate layer are defined by the following rules:

Each node (i.e. each factor) in the factor layer connects to the five nodes (i.e. five predicates) in the predicate layer which are distributed in its factor-space by weights 1, and connects to the other nodes by weights 0. For example, factor IC only connects to those nodes which are distributed in its factor-space as shown in Fig.-1.

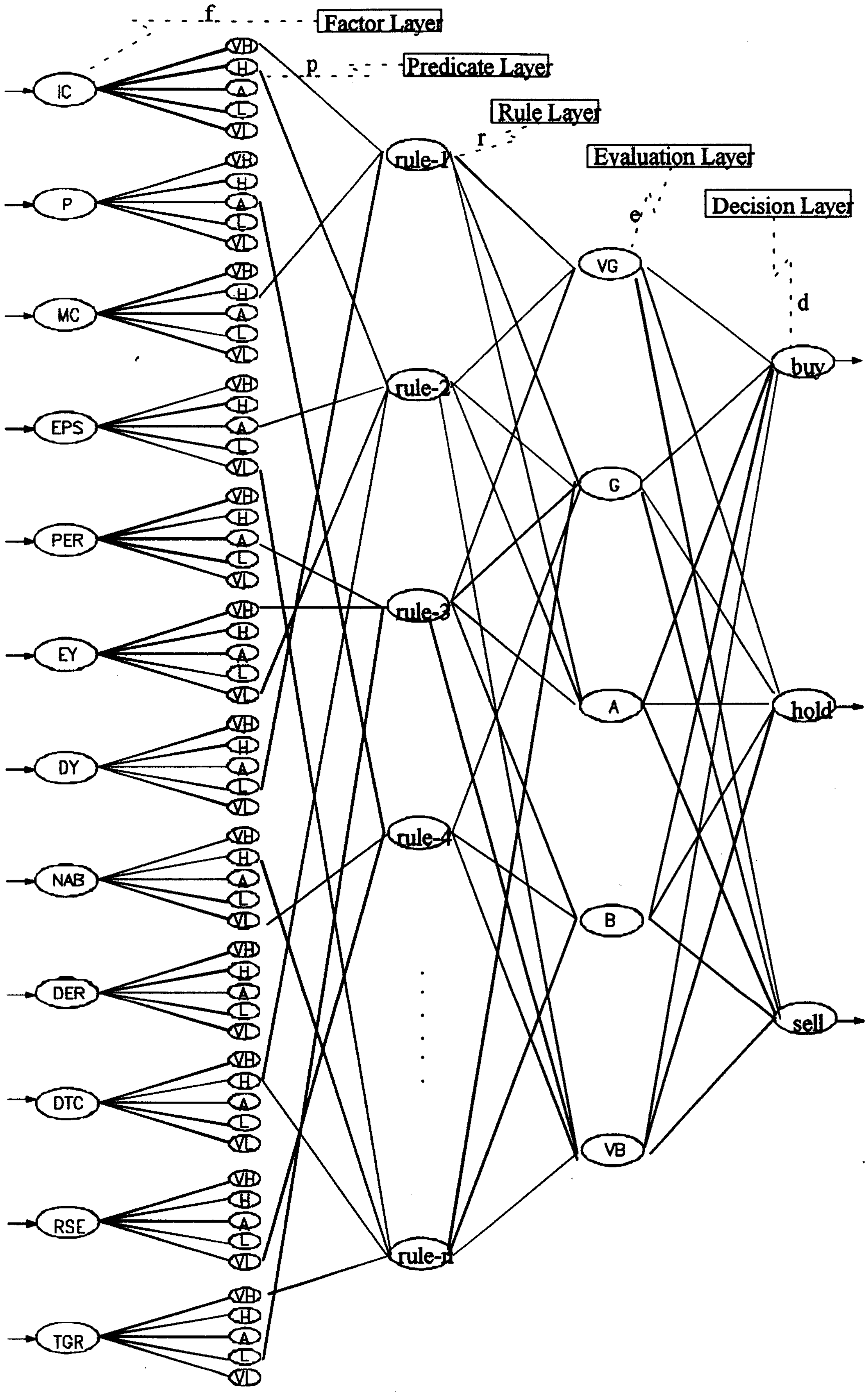


Figure 1. Decision-making Neural Networks

The weights between the predicate layer and the rule layer are defined by the following rules:

Because each node in the rule layer represents a rule, so each node (rule) in the rule layer only connects to those nodes (predicates) in the predicate layer which are related with this rule. Moreover, as inputs of a logic neural networks, these predicates connecting with the rule should be indicated in some sequence. So we use numbers {1, 2, 3, ...} as weights to denote the sequence of inputs. For those not connecting nodes, using 0 as their weights. For example, for Rule-1, we may have the weights shown in Fig.-4.

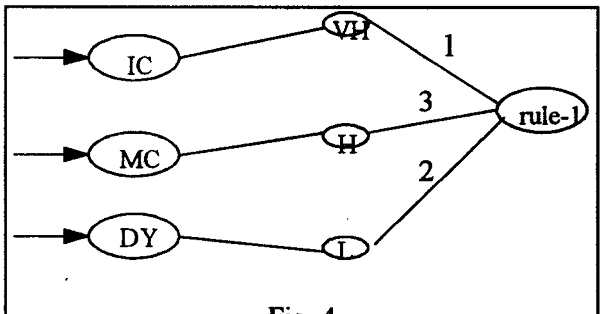


Fig. 4

The meaning of the Fig.-4 depends on the internal structure of rule-1.

The weights between the rule layer and evaluation layer are defined by the following rules:

For each rule, its seccedent is an evaluation "very good", "good", etc.. So each node (rule) in the rule layer connects to the node in the evaluation layer which is the seccedent of this rule by weight 1, and connects to the other nodes by weights 0. Further more, we can use possibility distribution in evaluation set {"very good", "good", "average", "bad", "very bad"} as a seccedent of rule. In this case, each node (rule) in the rule layer connects to the nodes in the evaluation layer with the weight values obtained from possibility distribution functions. For example, for Rule-1, suppose its seccedent is the possibility distribution function

$$\left\{\frac{0.9}{\text{very good}}, \frac{1}{\text{good}}, \frac{0.8}{\text{average}}, \frac{0.3}{\text{bad}}, \frac{0}{\text{very bad}}\right\}$$

then we can establish the weights between Rule-1 and all the nodes in the evaluation layer as shown in Fig.-5.

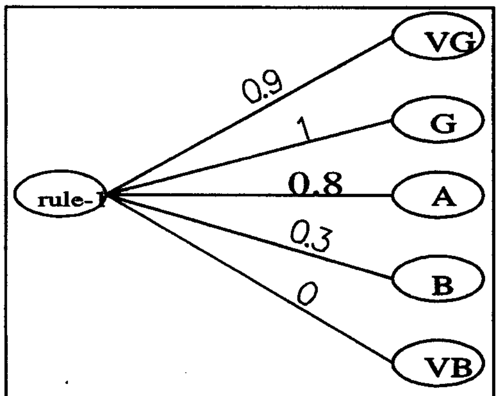


Fig. 5

The weights between the evaluation layer and the decision (output) layer is determined by experts according to their experiences. These weights can be modified when neural networks be training. For example, in the financial forecast field, we can use the weights in Fig.-6 as the initial weights.

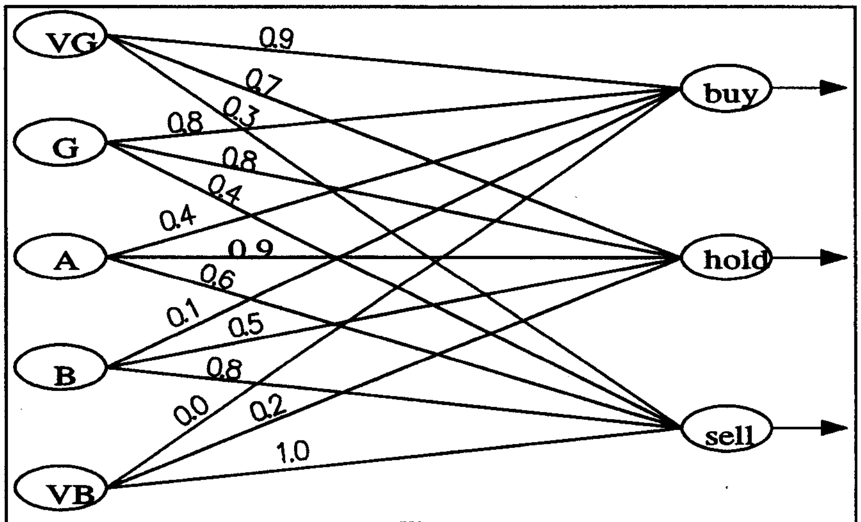
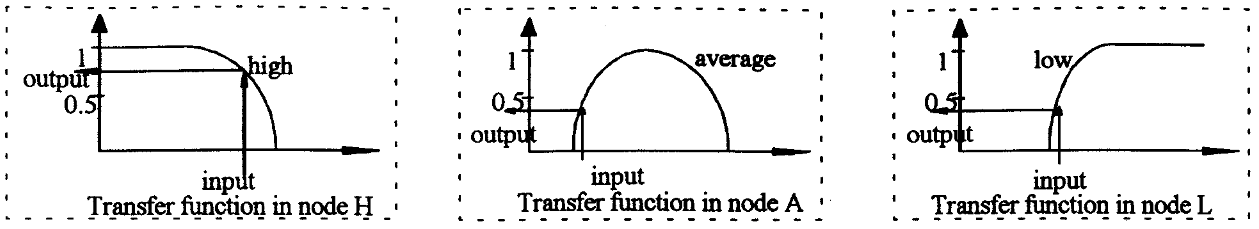


Fig. 6

So far we have built up the Decision-Support Neural Networks(DSNN) completely. In the following several sections, we will discuss the implementation of this neural networks, its learning capacity, the implementation of logic operations through networks, etc..

2. Implementation of DSNN:

In the input layer we input the values which indicate the states of all accounting items, and in the predicate layer these values are transferred to truth values through fuzzy predicates "very high", "high", "average", "low", "very low", and these truth values are output as the outputs of predicate layer. The fuzzy set representations of the three fuzzy predicates and transferring process of truth values in the predicate layer can be shown in the Fig.-7.



1
output
0.5
high
input
Transfer function in node H
1
0.5
output
average
input
Transfer function in node A
1
0.5
output
low
input
Transfer function in node L

Fig. 7

These three membership functions are the transfer functions in nodes H, A, and L respectively. The inputs are state values of items, and the outputs are truth values obtained from corespondent fuzzy membership function.

In the rule layer, each node represents the antecedent of a rule, and an antecedent is a combination of propositions trough logic operations, therefore each node in the rule layer is a logic combination of propositions. For example, if the rule is "if A and B both true, then C is true", then the node is prepositional combination formula $A \wedge B$; similarly, if the rule is "if A or B is true, and A and B can not be true at the same time, then C is true", then the node is logic formula $(A \vee B) \wedge \neg(A \wedge B)$. About what kind of the transfer functions we should take in the third layer and how to implement logic formulae by these transfer functions, we will discuss this problem in the section 4. There we will establish a new kind of function form, logic neural network function, to implement all forms of logic formulae. The outputs in the third layer are also logic truth values. In this layer, the transfer functions are logic neural networks, and the input values are obtained from formula

$$I_j = \mathrm{sgn}(W_{ij})O_i$$

the meanings of all symbols in this formula are stated in later part.

In the evaluation layer, the inputs are truth value which are output from rule layer, and its outputs are also truth values which are obtained by combination of all inputs from previous layer. This combination can be any one of the logic operations, such as min, product, boundary-product, etc..

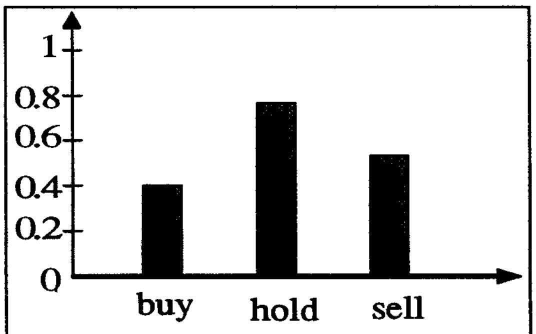


Fig. 8

In the output layer, the inputs and outputs are all logic truth values, and translation function is also any one of the logic operations, such as min, product, boundary-product, etc. In the output layer, all the outputs can form a possibility distribution function on the strategy set, this distribution function is just the basis of decision-making. According to this function, the action we should take is the one that has the maximum value in the distribution function. In the financial forecast field, if we choose {buy, hold, sell} as the strategy set, then the possibility distribution has the representation shown in Fig.-8, and according to this function we should take action "hold".

3. Learning of Membership Functions:

In this section, we will discuss the learning of fuzzy rules. The essential problem of learning fuzzy rules is learning fuzzy membership functions, and the essential problem of learning fuzzy membership functions is to set up a reasonable mathematical methods to modify membership function when the membership function is changed at one point. In the following, we will discuss the methods of modifying membership functions.

The general criteria of modifying membership functions:

① Modifying radius is positive proportional with the membership degree and the error δ;

② Considering left wing, right wing and median part (the set which have membership degree 1) separately.

③ Modified membership function should also be normal.

Using mathematical method, the above general criteria can be described as follows:

Definition: A normal fuzzy set $\tilde{A}$ is a fuzzy set on the real line that satisfies:

(1) exist at least one point $u_0 \in U$ such that $\tilde{A}(u_0) = 1$;

(2) as a function on the real line, $\tilde{A}$ is increasing on $(-\infty, u_0]$, and is decreasing on $[u_0, +\infty)$.

Before we continue our discussion, we first give several notations:

$\mathcal{F}_N(U)$ denotes the set of all normal fuzzy sets on the universe U, where U is a subset of the real line.

$$m^- \triangleq \inf\{u \mid \tilde{A}(u)=1\},\ m^+ \triangleq \sup\{u \mid \tilde{A}(u)=1\},\ A^- \triangleq (-\infty, m^-],\ A^+ \triangleq [m^+, +\infty),\ \overline{A} \triangleq [m^-, m^+].$$

$\tilde{A}_\lambda \triangleq \{x \mid x \in U, \tilde{A}(x) \geq \lambda\}$, and is called λ-cut set of $\tilde{A}$. In the case of $\tilde{A}$ is normal, $\tilde{A}_\lambda$ is an interval, and we denote $\tilde{A}_\lambda = [a_\lambda^-, a_\lambda^+]$, where $\lambda \in [0,1]$.

$\rho: [0,1] \times [0,1] \to [0, +\infty)$ is an increasing function with regard to its two variables respectively, and satisfies $\rho(y, 0) = 0$ for any $y \in [0,1]$.

$M: [0, +\infty) \to Z^+ = \{1, 2, 3, \cdots\cdots\}$ is an increasing function.

$\mathcal{U}(u)$ denotes any one neighbor of u in U, and $\mathcal{U}(u, r)$ denotes the neighbor of u with radius r, $\mathcal{U}^-(u, r)$ denotes the left half-neighbor of u with radius r, and $\mathcal{U}^+(u, r)$ denotes the right half-neighbor of u with radius r. They can be defined by the following formulae.

$$\mathcal{U}(u, r) \triangleq \{x \mid x \in U, |x-u| \leq r\},$$

$$\mathcal{U}^-(u, r) \triangleq \{x \mid x \in U, 0 \leq (u-x) \leq r\},$$

$$\mathcal{U}^+(u, r) \triangleq \{x \mid x \in U, 0 \leq (x-u) \leq r\}.$$

Definition: The bound pair function <u>bnd</u> on the real line **R** with α as the lower bound and with β as the upper bound is defined as follows:

$$\mathrm{bnd}(x)_\alpha^\beta \triangleq \begin{cases} \alpha & \text{if } x < \alpha; \\ \beta & \text{if } x > \beta; \\ x & \text{else.} \end{cases}$$

In particular, $\mathrm{bnd}_0^1(.)$ represents the bounded function which has the lower bound 0, and upper bound 1; $\mathrm{bnd}_{-1}^1(.)$ represents the bounded function which has the lower bound -1 and upper bound 1; etc. In order to simplicity, we usually use the concise form $\mathrm{bnd}(\cdot)$ to represent $\mathrm{bnd}_0^1(.)$.

Using u to denote the adjusted point, we will discuss the modifying membership function problem in three cases, $u \in A^-$, $u \in A^+$, and $u \in \overline{A}$.

● Case of $u \in A^-$.

Definition: Define mapping **Φ** as follows:

$$\Phi: \mathcal{F}_N(U) \times U \times [-1,1] \mapsto \mathcal{F}_N(U)$$

$$(\tilde{A}, u, \delta) \to \tilde{A}_{(u,\delta)}$$

where $\tilde{A}_{(u,\delta)}$ is defined as follows:

$$\tilde{A}_{(u,\delta)}(x) \triangleq \begin{cases} \xi^-(\tilde{A}, u, \delta)(x) & x \leq u; \\ \xi^+(\tilde{A}, u, \delta)(x) & u < x \leq m^-; \\ \varepsilon(\tilde{A}, u, \delta)(x) & m^- < x \leq (m^- + \theta(\tilde{A}, \kappa)); \\ \zeta^+(\tilde{A}, u, \delta)(x) & (m^- + \theta(\tilde{A}, \kappa)) < x \leq \tau; \\ \tilde{A}(x) & \text{else.} \end{cases}$$

where $\tau = m^- + \theta(\tilde{A}, \kappa) + \rho\big(\tilde{A}(m^- + \theta(\tilde{A}, \kappa)), (1 - \tilde{A}(m^- + \theta(\tilde{A}, \kappa)))\big)$, $\kappa = 1 - \xi^+(\tilde{A}, u, \delta)(m^-)$, and other functions are defined respectively by the following formulae.

$$\theta(\tilde{A},\kappa): \mathcal{F}_N(U)\times[0,1]\mapsto[0,+\infty)$$

is any one of the increasing functions with respect to κ when $\tilde{A}$ is fixed, and satisfies the condition: $\theta(\tilde{A},0)=0$ for any $\tilde{A}$.

$$\varepsilon(\tilde{A},u,\delta): [m^-, m^- + \theta(\tilde{A},\kappa)]\mapsto[0,1]$$

is any one of the increasing functions, and satisfies the conditions that $\varepsilon(\tilde{A},u,\delta)(m^-)=\xi^+(\tilde{A},u,\delta)(m^-)$ and $\varepsilon(\tilde{A},u,\delta)(m^- + \theta(\tilde{A},\kappa))=1$.

Suppose for any radius ρ, we divide the two intervals $[u-\rho, u]$ and $[u, u+\rho]$ to be $M(\rho)$ equal segments so that each section has the length $\frac{\rho}{M(\rho)}$, then we have the following function definitions.

Definition: Define function $\xi^-(\tilde{A},u,\delta)$ as follows:

When $x\in\mathcal{U}^-(u,0)=\mathcal{U}^-\left(u,\frac{0\times\rho}{M(\rho)}\right)$,

$$\xi^-(\tilde{A},u,\delta)(x)\triangleq\alpha_0^-\triangleq \mathrm{bnd}\left(\tilde{A}(x)+\delta\right)=\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-0)}{M(\rho)}\delta\right)$$

When $x\in\mathcal{U}^-\left(u,\frac{1\times\rho}{M(\rho)}\right)\setminus\mathcal{U}^-\left(u,\frac{0\times\rho}{M(\rho)}\right)$,

$$\xi^-(\tilde{A},u,\delta)(x)\triangleq\alpha_1^-\triangleq \min\left(\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-1)}{M(\rho)}\delta\right),\alpha_0^-\right)$$

In general situations, when $x\in\mathcal{U}^-\left(u,\frac{i\times\rho}{M(\rho)}\right)\setminus\mathcal{U}^-\left(u,\frac{(i-1)\times\rho}{M(\rho)}\right)$,

$$\xi^-(\tilde{A},u,\delta)(x)\triangleq\alpha_i^-\triangleq \min\left(\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-i)}{M(\rho)}\delta\right),\alpha_{i-1}^-\right)$$

where i = 1, 2, ..., $M(\rho)$.

Definition: Define function $\xi^+(\tilde{A},u,\delta)$ as follows:

When $x\in\mathcal{U}^+(u,0)\cap A^-=\mathcal{U}^+\left(u,\frac{0\times\rho}{M(\rho)}\right)\cap A^-$,

$$\xi^+(\tilde{A},u,\delta)(x)\triangleq\alpha_0^+\triangleq \mathrm{bnd}\left(\tilde{A}(x)+\delta\right)=\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-0)}{M(\rho)}\delta\right)$$

When $x\in\left(\mathcal{U}^+\left(u,\frac{1\times\rho}{M(\rho)}\right)\setminus\mathcal{U}^+\left(u,\frac{0\times\rho}{M(\rho)}\right)\right)\cap A^-$,

$$\xi^+(\tilde{A},u,\delta)(x)\triangleq\alpha_1^+\triangleq \max\left(\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-1)}{M(\rho)}\delta\right),\alpha_0^+\right)$$

In general situations, when $x\in\left(\mathcal{U}^+\left(u,\frac{i\times\rho}{M(\rho)}\right)\setminus\mathcal{U}^+\left(u,\frac{(i-1)\times\rho}{M(\rho)}\right)\right)\cap A^-$,

$$\xi^+(\tilde{A},u,\delta)(x)\triangleq\alpha_i^+\triangleq \max\left(\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-i)}{M(\rho)}\delta\right),\alpha_{i-1}^+\right)$$

where i = 1, 2, ..., $M(\rho)$.

Denote $\delta'=1-\tilde{A}(m^- + \theta(\tilde{A},\kappa))$, then $\zeta^+(\tilde{A},u,\delta)$ can be defined as follows:

Definition: Define function $\zeta^+(\tilde{A},u,\delta)$ as follows:

When $x \in \mathcal{U}^{+}\left(m^{-}+\theta(\tilde{A},\kappa),0\right)=\mathcal{U}^{+}\left(m^{-}+\theta(\tilde{A},\kappa),\frac{0\times\rho}{M(\rho)}\right)$,

$$\zeta^{+}(\tilde{A},u,\delta)(x) \triangleq \beta_0^{+} \triangleq \mathrm{bnd}\left(\tilde{A}(x)+\delta'\right)=\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-0)}{M(\rho)}\delta'\right)$$

When $x \in \mathcal{U}^{+}\left(m^{-}+\theta(\tilde{A}+\kappa),\frac{1\times\rho}{M(\rho)}\right)\backslash\,\mathcal{U}^{+}\left(m^{-}+\theta(\tilde{A}+\kappa),\frac{0\times\rho}{M(\rho)}\right)$,

$$\zeta^{+}(\tilde{A},u,\delta)(x) \triangleq \beta_1^{+} \triangleq \min\left(\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-1)}{M(\rho)}\delta'\right),\beta_0^{+}\right)$$

In general situations, when

$x \in \mathcal{U}^{+}\left(m^{-}+\theta(\tilde{A}+\kappa),\frac{i\times\rho}{M(\rho)}\right)\backslash\,\mathcal{U}^{+}\left(m^{-}+\theta(\tilde{A}+\kappa),\frac{(i-1)\times\rho}{M(\rho)}\right)$,

$$\zeta^{+}(\tilde{A},u,\delta)(x) \triangleq \beta_i^{+} \triangleq \min\left(\mathrm{bnd}\left(\tilde{A}(x)+\frac{(M(\rho)-i)}{M(\rho)}\delta'\right),\beta_{i-1}^{+}\right)$$

where $i = 1, 2, \ldots, M(\rho)$.

All above procedures can be shown in the following Fig.-9.

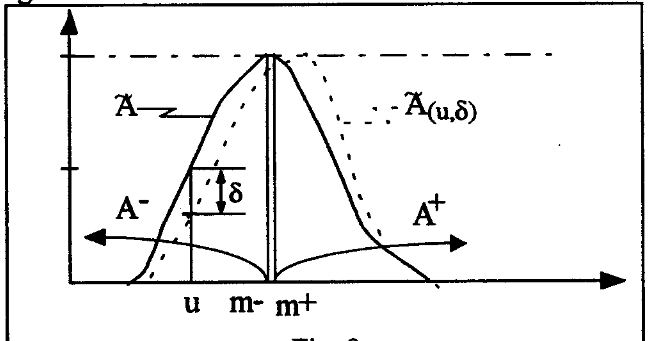


Fig. 9

Theorem: The definition of membership function $\tilde{A}_{(u,\delta)}$ is well defined. That is to say that we can get one and only one membership function according to the above definitions, and this membership function is normal.

● Case of $u \in A^{+}$.

When the adjusted point u is in A^{+}, we can similarly discuss this problem, and have a correspondent result.

● Case of $u \in \overline{A}$.

When the adjusted point u is in $\overline{A}$, we have $u \in \tilde{A}_{1-\delta}=[a_{1-\delta}^{-},a_{1-\delta}^{+}]$, and we may discuss this problem according to the distances between u and $a_{1-\delta}^{-}$, $a_{1-\delta}^{+}$.

In the case of $\left|u-a_{1-\delta}^{-}\right|<\left|a_{1-\delta}^{+}-u\right|$, we may define

$$\tilde{A}_{(u,\delta)}(x) \triangleq \begin{cases} \xi^{-}(\tilde{A},u,\delta)(x) & x \le u; \\ \varepsilon(\tilde{A},u,\delta)(x) & u < x \le (u+\theta(\tilde{A},\kappa)); \\ \zeta^{+}(\tilde{A},u,\delta)(x) & (u+\theta(\tilde{A},\kappa)) < x \le \tau; \\ \tilde{A}(x) & \text{else.} \end{cases}$$

where $\tau = u+\theta(\tilde{A},\kappa)+\rho\left(\tilde{A}(u+\theta(\tilde{A},\kappa)),(1-\tilde{A}(u+\theta(\tilde{A},\kappa)))\right)$, $\kappa = 1-\xi^{-}(\tilde{A},u,\delta)(u)$, and other functions have the same meanings as in the previous definitions.

In the case of $\left|u-a_{1-\delta}^{-}\right|>\left|a_{1-\delta}^{+}-u\right|$, we have a corresponding result.

In the case of $\left|u-a_{1-\delta}^{-}\right|=\left|a_{1-\delta}^{+}-u\right|$, we had better take observing further, so we do not take any modification in this moment.

4. The implementation of the logic formula:

The representation of rules is one of the most important subjects in AI and decision-making science. A good representation can enable us to compute formulae and execute rules easily and feasibly. On the other hand, the problem of representing rule is acutely the same problem of representing the logic formula which is the antecedent of the rule. That is to say that in order to represent the rule "if A and B are both true, then the stock is good", we acutely only need to represent the logic formula $A \wedge B$. Based on the truth value flow inference[2], in the following we will introduce a type of networks to represent logic formulae.

We first discuss the representation of formulae only with operators $\wedge$ and $\neg$.

Logic formula A, ¬A, A∧B, and A∧¬B have the following network representations respectively:

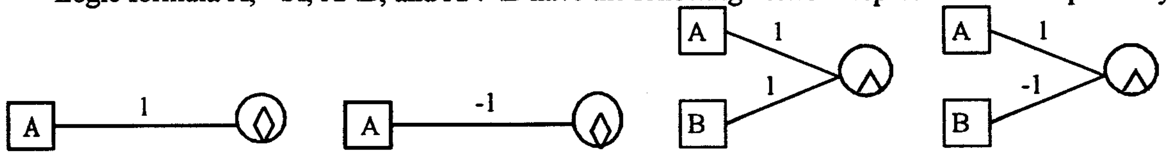


Fig. 10

If the rule is "A is true with truth value s, then the stock is good", or the rule is "B is false with truth value t, then the stock is bad", or the rule is "A is true with truth value s and B is true with truth value t, then the stock is very good", or the rule is "A is true with truth value s and B is false with truth value t, then the stock is average", then the logic formulae corresponding to these rules are A[s], ¬B[t], A[s]∧B[t], and A[s]∧¬B[t], and the Fig.-10 will be generalized to the Fig.-11.

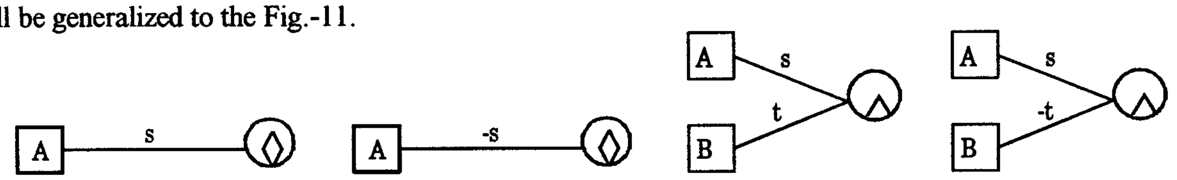


Fig. 11

Let s and t range in the interval [-1, 1], then the eight representations in Fig.-10 and Fig.-11 in the above can be integrated into the Fig.-12 as shown in the following:

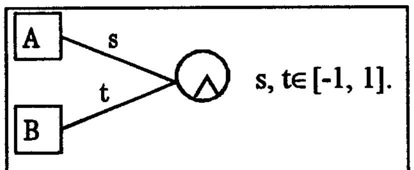


Fig. 12

When t=0, ◊ replaces ∧, and Fig.-12 becomes the first two representations in Fig.-11.

Besides the boundary function bnd given before, here we also use sgn(x), int(x), and abs(x) to denote the sign function, the integral function, and absolute function respectively. And we use dec(x) to denote the decimal function which is defined by the formula:

$$\mathrm{dec}(x) \overset{\Delta}{=} x - \mathrm{int}(x)$$

Using T(A) and T(B) to denote the truth values of A and B, and using in(·) and out(·) to denote the input and output values of each node in networks respectively, then, in reference to Fig.-12, the input and output in 'node A' are both defined as T(A), i.e., in(A)=out(A)=T(A); The input and output in 'node B' are both defined as T(B), i.e., in(B)=out(B)=T(B); The inputs in 'node ∧' are T(A) and T(B), i.e., $\mathrm{in}_1(\wedge) = T(A)$, $\mathrm{in}_2(\wedge) = T(B)$, and the output value $\mathrm{out}(\wedge)$ in 'node ∧' is defined as

$$\mathrm{out}(\wedge) \overset{\Delta}{=} \mathrm{bnd}\left(\frac{\mathrm{dec}(\mathrm{sgn}(s)\,\mathrm{in}_1(\wedge))}{\mathrm{abs}(s)}\right) \otimes \mathrm{bnd}\left(\frac{\mathrm{dec}(\mathrm{sgn}(t)\,\mathrm{in}_2(\wedge))}{\mathrm{abs}(t)}\right) \quad \cdots\cdots \quad \text{(Formula-I)}$$

where ⊗ could be any one operator in the operator set {min, •, ×, ∧, bounded-product, ...}. When t=0, we define $\mathrm{bnd}\left(\frac{\mathrm{dec}(\mathrm{sgn}(t)T(A))}{\mathrm{abs}(t)}\right) = 1$, and use ◊ to replace ∧, then, in this case,

$\mathrm{out}(\Diamond) = \mathrm{bnd}\left(\frac{\mathrm{dec}(\mathrm{sgn}(s)T(A))}{\mathrm{abs}(s)}\right)$, this is the output value of 'node ◊' in Fig.-11.

For those logic formulae only with operators ∨ and ¬, we have a similar result. That is to say that, for logic formulae $A[s] \vee B[t]$, $A[s] \vee \neg B[t]$, etc., we have the networks representation as shown in Fig.-13.

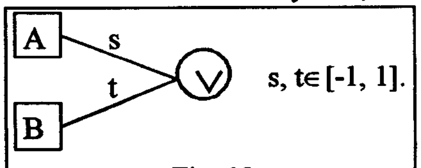


Fig. 13

In Fig.-13, the inputs and outputs in nodes A and B have the same values with those in Fig.-12, the inputs in 'node ∨' are T(A) and T(B), i.e.,

$\mathrm{in}_1(\vee) = T(A)$, $\mathrm{in}_2(\vee) = T(B)$, and the output value $\mathrm{out}(\vee)$ in 'node $\vee$' is defined as

$$\mathrm{out}(\vee) \triangleq \mathrm{bnd}\left(\frac{\mathrm{dec}(\mathrm{sgn}(s)\,\mathrm{in}_1(\vee))}{\mathrm{abs}(s)}\right) \oplus \mathrm{bnd}\left(\frac{\mathrm{dec}(\mathrm{sgn}(t)\,\mathrm{in}_2(\vee))}{\mathrm{abs}(t)}\right) \qquad \cdots\cdots\text{(Formula-II)}$$

where $\oplus$ could be any one operator in operator set {max, +, $\vee$, bounded-sum, ...}. When t=0, we define $\mathrm{bnd}\left(\frac{\mathrm{dec}(\mathrm{sgn}(t)T(A))}{\mathrm{abs}(t)}\right) = 0$, and use $\lozenge$ to replace $\vee$.

For logic operator $\rightarrow$, we use $(A \wedge \neg A) \vee (A \wedge B)$ to replace $A \rightarrow B$ in logic formulae.

Here we give an example of the neural networks representation of logic formula. The logic formula

$$(((A_1\wedge A_2)\vee(A_1\wedge A_3)\vee(A_1\wedge A_4)\vee(A_2\wedge A_3)\vee(A_2\wedge A_4)\vee(A_3\wedge A_4))[0.8]\wedge$$
$$((A_1\wedge A_2\wedge A_3)\vee(A_1\wedge A_2\wedge A_4)\vee(A_1\wedge A_3\wedge A_4)\vee(A_2\wedge A_3\wedge A_4))[0.5])\vee$$
$$(\neg((\neg A_1\vee\neg A_2\vee\neg A_3)\wedge(\neg A_1\vee\neg A_2\vee\neg A_4)\wedge(\neg A_1\vee\neg A_3\vee\neg A_4)\wedge$$
$$(\neg A_2\vee\neg A_3\vee\neg A_4)))[0.7]$$

has the networks representation as shown in Fig.-14.

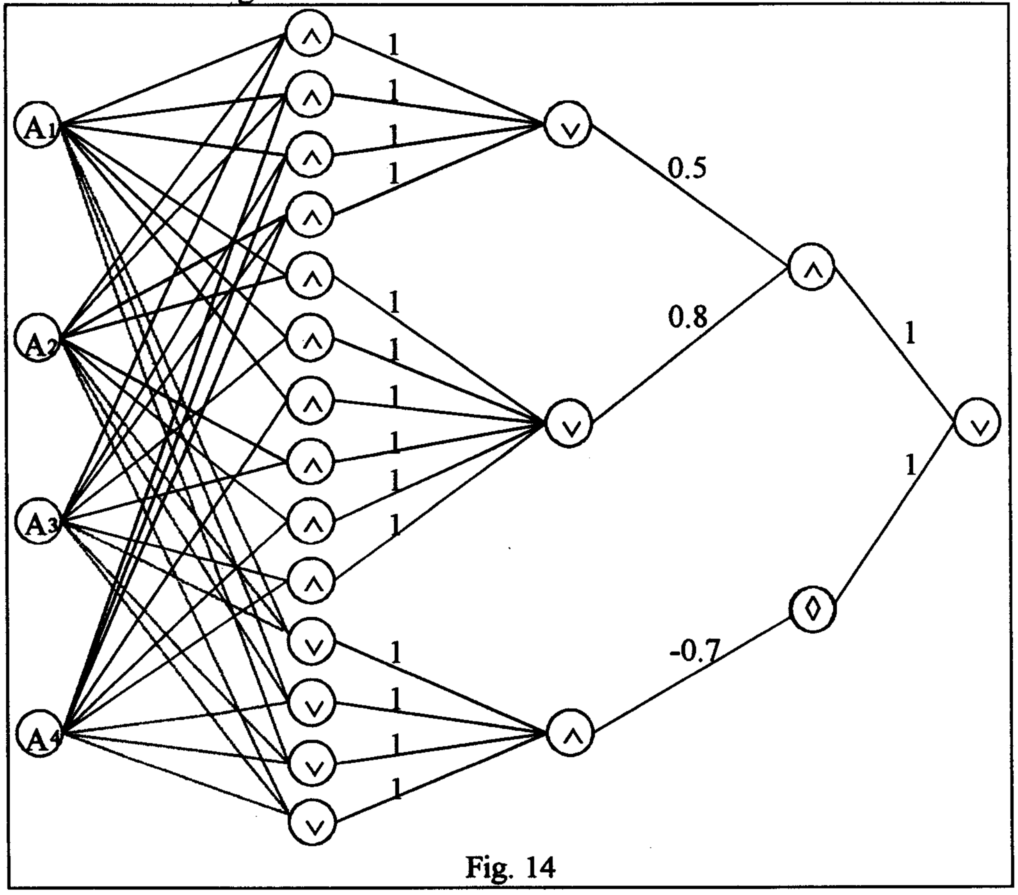


Fig. 14

Through this logic neural networks, as long as we input the truth values of atomic propositions A,B,C, ..., we can get the truth value of any logic formula.

The learning rules of logic neural networks:

Here we suppose that W(i:j, i+1:k) represents the weight between the j-th node in layer i and the k-th node in layer i+1, and use $\Delta W(i:j, i+1:k)$ to denote its difference; and suppose that I(i, j), O(i, j) and E(i, j) to denote the input, output and error of the j-th node in layer i respectively; and use T to denote the target value; then we may define E(i, j) and $\Delta W(i:j, i+1:k)$ as follows:

In the output layer, $E(i, j) = T - O(i, j)$

in the other layer,

$$E(i, j) = \sum_k W(i:j, i+1:k)E(i+1, k)$$

$$\Delta W(i:j, i+1:k) = -\alpha E(i+1, k)$$

where α is proper positive number parameter.

5. The learning capacity of this neural networks:

This DSNN has two aspects of learning capacities. The first learning capacity is learning weights, and the second learning capacity is learning fuzzy rules. Through these learnings, we can get proper weights to express the connections between nodes and proper fuzzy rules. In the previous section, we have discussed the learning problem of membership functions, i.e. learning problem of fuzzy rules. In the following we will discuss the learning rules of weights.

In order to write clearly, we first give some symbols and their meanings:

❶ Using f, p, r, e, and d to denote the Factor Layer (f), Predicate Layer (p), Rule Layer (r), Evaluation Layer (e), and Decision Layer (d) respectively; And using L to denote the character set {f, p, r, e, d}, i.e., $L \overset{\Delta}{=} \{f, p, r, e, d\}$; And using Z+ to denote the integer set $\{0, 1, 2, 3, \cdots\cdots\}$.

❷ Using U(c) to denote the number of nodes (units) in layer c, where $c \in L$; And using U(c) to denote the integer set {0, 1, 2, ..., U(c)-1} in the same time.

❸ Using N(c, i) to denote the i-th node in layer c, and using I(c, i) and O(c, i) to denote the input and output values in node N(c, i), where $c \in L$, $i \in U(c)$.

❹ Using $W(c_1{:}i, c_2{:}j)$ to denote the weight between node $N(c_1, i)$ and node $N(c_2, j)$, and using $\Delta W(c_1{:}i, c_2{:}j)$ to denote the difference of $W(c_1{:}i, c_2{:}j)$, where $c_1, c_2 \in L$, and $i \in U(c_1)$, $j \in U(c_2)$.

❺ Using T(i) ($i \in U(d)$) to denote the target values.

❻ Using E(c, i) to denote the error in node N(c, i), where $c \in L$, $i \in U(c)$.

The learning rules of DSNN are given as follows:

Step 1. Set the errors $E(d,i) = (T(i) - O(d,i))$, where $i \in U(d)$.

Step 2. Set the errors E(e, i) as the following formula:

$$E(e,i) = \frac{\sum\limits_{j \in U(d)} E(d,j)W(e{:}i, d{:}j)}{U(d)}, \quad \text{where } i \in U(e).$$

Step 3. Set the errors E(r, i) and ΔW(r:i, e:j) as the following formula:

$$E(r,i) = \frac{1}{U(e)} \sum_{j \in U(e)} \text{bnd}_{-1}^{1}\left(\frac{O(r,i)E(e,j)}{O(e,j)} W(r{:}i, e{:}j) \right)$$

$$\Delta W(r{:}i, e{:}j) = \text{bnd}_{-1}^{1}\left(\frac{E(e,j)}{O(e,j)} W(r{:}i, e{:}j) \right)$$

Step 4. Using E(r, i) ($i \in U(r)$) as the errors in logic neural networks and according to the learning rules of logic neural networks we describe previously we can get E(p, i) ($i \in U(p)$).

Step 5. The problem of error modifications in layer p is just the learning problem of membership functions which we have discussed in above.

REFERENCES

[1] F.S. Wong and P.Z. Wang, A Stock Selection Strategy Using Fuzzy Neural Networks, Neurocomputing 2 (1990/91) 233-242.

[2] P.Z. Wang, etc., Truth-valued-flow inference, BUSEFAL No 38 (Spring 1989) 130-139.

[3] P.Z. Wang, Random sets and fuzzy sets, International Encyclopedia on Systems and Control, Pergamon Press (1987).

[4] P.Z. Wang, Fuzzy Sets Theory and Its Applications, Shanghai Scientific & Technical Publishers, Shanghai, 1983.

[5] X.H. Zhang, Francis Wong, H.C. Lui, P.Z. Wang, Theoretical basis of truth value flow inference and its applications, Proceedings of the First Singapore International Conference on Intelligent Systems, September 28-October 1, 1992.

[6] X.H. Zhang, H.C. Lui, Francis Wong, Z.L. Shen, The Coupling of Truth Value Flow Inference Neural Networks and Approximate Reasoning Based on Similarity Theory, Proceedings of the Second IEEE International Conference on Fuzzy Systems, March 28- April 1, 1993, San Francisco, California (*to appear*).

[7] Tarun Khanna, Foundations of Neural Networks, Addison-Wesley Publishing Company, 1990.

[8] P. L. Pau, Fuzzy Patter Recognitions and Neural Networks, 1991.

[9] L.A. Zadeh, PRUF-a meaning representation language for natural languages, Fuzzy Reasoning and its Applications (pp. 1-66), *edited by* Dr. E.H. Mamdani and Professor B.R. Gaines, Academic Press, 1981.