The Introduction of AFS Theory and Its Matlab Codes

Editted by Xiaolei Zhu, Xuelian Xu, Xiaodong Liu

Abstract

In this paper, we introduce the basic ideas and the results of AFS (Axiomatic Fuzzy Set) theory with some simple illustrative examples. All operations and the calculations for AFS theory could be implemented by the Matlab programs and functions whose applied methods are explained and original codes are listed in the Appendix.

1. *AFS algebras*

In this section, we recall some notations and properties of AFS theory that will be of immediate use in the study. For more details the reader is referred to [7-13]

In [7] defined was a family of molecular lattices, the AFS algebras, denoted as *EI*, *EII*, ..., EI^n, $E^{\#}I$, algebras. Next these AFS algebras were applied to study the lattice valued representations of fuzzy concepts. The following example serves as an introductory illustration of the AFS algebra.

Example 1. Let $X=\{x_1, x_2, \ldots., x_{10}\}$ be a set of 10 people and their features (attributes) which are described by real numbers (age, height, weight, salary, estate), Boolean values (gender) and the order relations (hair black, hair white, hair yellow); see Table 1.

Table 1-Descriptions of features

		appearance		wealth		gender		hair color		
	age	height	weight	salary	estate	male	female	black	white	yellow
x_1	20	1.9	90	1	0	1	0	6	1	4
x_2	13	1.2	32	0	0	0	1	4	3	1
x_3	50	1.7	67	140	34	0	1	6	1	4
x_4	80	1.8	73	20	80	1	0	3	4	2
x_5	34	1.4	54	15	2	1	0	5	2	2
x_6	37	1.6	80	80	28	0	1	6	1	4
x_7	45	1.7	78	268	90	1	0	1	6	4
x_8	70	1.65	70	30	45	1	0	3	4	2
x_9	60	1.82	83	25	98	0	1	4	3	1
x_{10}	3	1.1	21	0	0	0	1	2	5	3

Here the number *i* in the "*hair color*" columns which corresponds to some $x \in X$ implies that the hair color of *x* has ordered following our perception of the *i*th color. For example, the numbers in the column *"hair black"* imply some order (>)

$$x_7 > x_{10} > x_4 = x_8 > x_2 = x_9 > x_5 > x_6 = x_3 = x_1$$

When moving from right to left, the relationship states how strongly the hair color under consideration resembles black color. In this order, $x_i > x_j$ (e.g., $x_7 > x_{10}$) states that the hair of x_i is closer to the black color than the color of hair the individual x_j. The relationship $x_i = x_j$ (e.g., $x_4 = x_8$) means that the hair of x_i looks as black as the one of x_j. A concept on *X* may associate to one or more features. For instance, the fuzzy concept *"tall"* associates a single feature *"height"* and the

fuzzy concept *"old white hair males"* associates three features *"age", "hair color_black"* and *"gender_male"*. Many concepts may associate with a single feature. For instance, the fuzzy concepts *"old", "young" and "about 40 years old"* all associate to feature *"age"*. Let $M=\{m_1, m_2, \ldots, m_{12}\}$ be the set of fuzzy or Boolean concepts on X and each $m\in M$ associates to a single feature. Where m_1: *"old persons"*, m_2: *"tall persons"*, m_3: *"heavy persons"*, m_4: *"high salary"*, m_5: *"more estate"*, m_6: *"male"*, m_7: *"female"*, m_8: *"black hair persons"*, m_9: *"white hair persons"*, m_{10}: *"yellow hair persons"*, m_{11}: *"young persons*", m_{12}: *"the persons about 40 years old*". The elements of M are viewed as "elementary" (or " simple") concepts. For each set of concepts $A\subseteq M$, $\prod_{m\in A}m$ represents conjunction of the concepts in A. For instance, $A=\{m_1, m_6\}\subseteq M$, $\prod_{m\in A}m = m_1m_6$ representing a new fuzzy concept *"old males"* which is associating to features age and gender. For $\sum_{i\in I}(\prod_{m\in Ai} m)$, which is a formal sum of $\prod_{m\in Ai} m$, $A_i\subseteq M$, $i\in I$, is the disjunction of the conjunctions represented by $\prod_{m\in Ai}m$'s (i.e., the disjunctive normal form of a formula representing a concept). For example, we may have $\gamma=m_1m_6+m_1m_3+m_2$ which translates as *"old males"* or *"heavy old persons"* or *"tall persons"*. (the "+" denotes here a disjunction of concepts). While M may be a set of fuzzy or Boolean (two-valued) concepts, every $\sum_{i\in I}(\prod_{m\in Ai} m)$, $A_i\subseteq M$, $i\in I$, has a well-defined meaning such as the one we have discussed above. By a straightforward comparison of the expressions

$$m_3m_8+m_1m_4+m_1m_6m_7+m_1m_4m_8 \text{ and } m_3m_8+m_1m_4+m_1m_6m_7,$$

we conclude that their left side and right sides are equivalent. Considering the terms on the left side of the expression, for any x, the degree of x belonging to the fuzzy concept represented by $m_1m_4m_8$ is always less than or equal to the degree of x belonging to the fuzzy concept representing by m_1m_4. Therefore, the term $m_1m_4m_8$ is redundant when forming the left side of the fuzzy concept. Let us take into consideration two expressions of the form α: $m_1m_4+m_2m_5m_6$ and ν: $m_5m_6+m_5m_8$. The semantic content of the fuzzy concepts "α or ν" and "α and ν" can be expressed as follows

"α or ν": $m_1m_4+m_2m_5m_6+m_5m_6+m_5m_8$ equivalent to $m_1m_4+m_5m_6+m_5m_8$,

"α and ν": $m_1m_4m_5m_6+m_2m_5m_6+m_1m_4m_5m_8+m_2m_5m_6m_8$ equivalent to

$$m_1m_4m_5m_6+m_2m_5m_6+m_1m_4m_5m_8.$$

The semantics of the logic expressions such as "*equivalent to*", "*or*" and "*and*" as expressed by $\sum_{i\in I}(\prod_{m\in Ai} m)$, $A_i\subseteq M$, $i\in I$, can be formulated in terms of the AFS algebra in the following manner.

It is known [3] that a lattice is a partially ordered set L in which any two elements $a, b \in L$ have a least upper-bound (i.e., $a\vee b$) and a greatest lower bound (i.e., $a\wedge b$). A partially ordered set L is called a complete lattice if every subset $A\subseteq L$ has a *sup* and an *inf*, denoted by $\vee_{a\in A}a$ and $\wedge_{a\in A}a$ respectively. A complete lattice is called a completely distributive lattices (or molecular lattice [3] if one of the conditions shown below (CD1 or CD2) holds

$$(CD1)\ \bigwedge_{i\in I}\Big(\bigvee_{j\in J_i} a_{ij}\Big) = \bigvee_{f\in\prod_{i\in I} J_i}\Big(\bigwedge_{i\in I} a_{if(i)}\Big),$$

$$(CD2)\ \bigvee_{i\in I}\Big(\bigwedge_{j\in J_i} a_{ij}\Big) = \bigwedge_{f\in\prod_{i\in I} J_i}\Big(\bigvee_{i\in I} a_{if(i)}\Big)$$

where $\forall\, i\in I$, $\forall\, j\in J_i$, $a_{ij}\in L$, and $f\in\Pi_{i\in I}J_i$ means that f is a mapping f: $I\to \cup_{i\in I}J_i$ such that $f(i)\in J_i$ for any $i\in I$.

Let M be a non-empty set. The set EM^* is defined by

$EM^*=\{ \sum_{i\in I}(\prod_{m\in Ai} m) \mid A_i\subseteq M, i\in I, I \text{ is a non-empty indexing set } \}$.

Matlab application methods 1:

The elements of EM are represented by matrices. For example*, $M=\{m_1, m_2, \ldots, m_{12}\}$ *in Example* 1. *For* $\eta=m_1m_4m_8$, $\zeta=m_3m_8+m_1m_4+m_1m_6m_7+m_1m_4m_8\in EM^*$. *Matrices* U=[0 1 4 8]', V=[0 0 3 8; 0 0 1 4; 0 1 6 7; 0 1 4 8]', *where matrices U,V represent* η, ζ.

Definition 1. ([7]) Let M be a non-empty set. A binary relation R on EM^* is defined as follows. For $\sum_{i\in I}(\prod_{m\in Ai} m)$, $\sum_{i\in J}(\prod_{m\in Bj} m) \in EM^*$,
$[\sum_{i\in I}(\prod_{m\in Ai} m)]R[\sum_{i\in J}(\prod_{m\in Bj} m)]\Leftrightarrow$ (i) $\forall A_i$ ($i\in I$), $\exists B_h$ ($h\in J$) such that $A_i\supseteq B_h$; (ii) $\forall B_j$ ($j\in J$), $\exists A_k$ ($k\in I$) such that $B_j\supseteq A_k$.

It is clear that R is an equivalence relation. The quotient set EM^*/R is denoted by EM. The notation $\sum_{i\in I}(\prod_{m\in Ai}m)=\sum_{i\in J}(\prod_{m\in Bj}m)$ means that $\sum_{i\in I}(\prod_{m\in Ai}m)$ and $\sum_{i\in J}(\prod_{m\in Bj}m)$ are equivalent under equivalence relation R. Thus the semantics they represent are equivalent. In Example 1, for $\xi=m_3m_8+m_1m_4+m_1m_6m_7+m_1m_4m_8$, $\zeta=m_3m_8+m_1m_4+m_1m_6m_7\in EM$, by Definition 1 we have $\xi=\zeta$. In what follows, each $\sum_{i\in I}(\prod_{m\in Ai} m)\in EM$ is called a fuzzy concept.

Matlab application methods 2:

We employ the function: equal_EI(*M,N*) *to judge whether two fuzzy concepts in EM* represented by matrices M and N are equivalent*. *For example*, $M=\{m_1, m_2, \ldots, m_{12}\}$ *in Example* 1. *For fuzzy concept* $\xi=m_3m_8+m_1m_4+m_1m_6m_7+ m_1m_4m_8$, $\zeta=m_3m_8+m_1m_4+m_1m_6m_7\in EM^*$. *Matrices* U=[0 0 3 8; 0 0 1 4; 0 1 6 7; 0 1 4 8]', V=[0 0 3 8; 0 0 1 4; 0 1 6 7]' *represent the fuzzy concepts* η, ζ *respectively*.

f=equal_EI(*U*,*V*)

f=1

We conclude that η *and* ζ *are equivalent*.

In general, we employ functions: reduceEI(*M*) *to reduce a fuzzy concept represented by matrix M.*

f=reduceEI(*U*)

f=[0 0 3 8; 0 0 1 4; 0 1 6 7]' %*representing the fuzzy concept* $m_3m_8+m_1m_4+ m_1m_6m_7$.

Theorem 1. ([7]) Let M be a non-empty set. Then (EM, $\vee$, $\wedge$) forms a completely distributive lattice under the binary compositions $\vee$ and $\wedge$ defined as follows. For any $\sum_{i\in I}(\prod_{m\in Ai} m)$, $\sum_{i\in J}(\prod_{m\in Bj} m) \in EM^*$

$$[\sum_{i\in I}(\prod_{m\in Ai} m)] \vee [\sum_{i\in J}(\prod_{m\in Bj} m)] = \sum_{k\in I\dot\cup J}(\prod_{m\in Ck} m) \quad (1)$$

$$[\sum_{i\in I}(\prod_{m\in Ai}m)] \wedge [\sum_{i\in J}(\prod_{m\in Bj}m)] = \sum_{i\in I,\ j\in J}(\prod_{m\in Ai\cup Bj} m) \quad (2)$$

where for any $k\in I\dot\cup J$ (the disjoint union of I and J, i.e., every element in I and every element in J are always regarded as different elements in $I\dot\cup J$), $C_k=A_k$, if $k\in I$, and $C_k=B_k$, if $k\in J$.

In Example 1, for $\alpha=m_1m_4+m_2m_5m_6$, $\nu=m_5m_6+m_5m_8\in EM$, in virtue (1) and (2) we know that

the semantic content of the fuzzy concepts "α or ν" and "α and ν" are expressed as $\alpha\vee\nu$ and $\alpha\wedge\nu$, respectively. The algebraic operations realized on them come in the form:

$$\alpha\vee\nu = m_1m_4+m_2m_5m_6+m_5m_6+m_5m_8 = m_1m_4+m_5m_6+m_5m_8,$$
$$\alpha\wedge\nu = m_1m_4m_5m_6+m_2m_5m_6+m_1m_4m_5m_8+m_2m_5m_6m_8$$
$$= m_1m_4m_5m_6+m_2m_5m_6+m_1m_4m_5m_8.$$

(EM, $\vee$, $\wedge$) is called the EI (expanding one set M) algebra over M, one type of AFS algebra. For $\alpha=\sum_{i\in I}(\prod_{m\in Ai}m)$, $\beta=\sum_{j\in J}(\prod_{m\in Bj}m)\in EM$, $\alpha\leq\beta\Leftrightarrow\alpha\vee\beta=\beta\Leftrightarrow\forall A_i\ (i\in I)$, $\exists B_h\ (h\in J)$ such that $A_i\supseteq B_h$. For example, $\alpha=m_1m_3m_4+m_3m_2+m_2m_6m_8+m_1m_4m_5m_8$, $\beta=m_1m_3+m_2+m_8$.

Matlab application methods 3:

We employ functions: EIsum(*M*,*N*), EImult(*M*,*N*) *to execute the binary compositions* $\vee$ *and* $\wedge$ *on concepts represented by matrices M,N. For example*, $M=\{m_1, m_2, \ldots, m_{12}\}$ *in Example* 1. *For fuzzy concepts* $\alpha=m_1m_3m_4+m_3m_2+m_2m_6m_8+m_1m_4m_5m_8$, $\beta=m_1m_3+m_2+m_8\in EM^*$. *U*=[0 0 1 3 4 ;0 0 0 3 2; 0 0 2 6 8; 0 1 4 5 8]', *V*=[0 1 3; 0 0 2; 0 0 8]', *where matrices U,V represent the fuzzy concepts* α, β.

f=EIsum(*U*,*V*)

f=[0 1 3; 0 0 2; 0 0 8]' % *representing the fuzzy concept* $m_1m_3+m_2+m_8$.

f=EImult(*U*,*V*)

f=[0 0 1 3 4; 0 0 0 2 3; 0 0 2 6 8; 0 1 4 5 8]' % *representing the fuzzy concept* $m_1m_3m_4+m_2m_3+m_2m_6m_8+m_1m_4m_5m_8$.

We employ the function less_EI(*M*,*N*) *to judge whether a fuzzy concept represented by matrix M is less than that represented by N*.

f=less_EI(*U*,*V*)

f=1

We conclude that concept α *is less than* β.

In [10], the authors proved that the operator " $'$ " is an order-reversing involution of EI algebra EM, if for any $\sum_{i\in I}(\prod_{m\in Ai}m)\in EM$,

$$(\sum_{i\in I}(\prod_{m\in Ai}m))'=\wedge_{i\in I}(\vee_{m\in Ai}m')=\wedge_{i\in I}(\sum_{m\in Ai}m') \quad (3)$$

If m' stands for the negation of the concept $m\in M$, then for any fuzzy concept $\zeta\in EM$, ζ' means the logical negation of ζ. In Example 1, for $\gamma=m_1m_6+m_1m_3+m_2\in EM$

$$\gamma' = (m_1m_6+m_1m_3+m_2)'$$
$$= (m_1'+m_6')\wedge(m_1'+m_3')\wedge m_2'$$
$$= (m_1'+m_6'm_3')\wedge m_2'$$
$$= m_1'm_2'+m_2'm_3'm_6'$$

γ', which is the logical negation of $\gamma=m_1m_6+m_1m_3+m_2$, reads as *"not old and not tall persons"* or *"not tall and not heavy females"*.

The algebra system (EM, $\wedge$, $\vee$, $'$) as a lattice not only provides a sound mathematical tool for us to study and determine the upper and lower approximations of a fuzzy set, but also ensures that they are the approximations of the fuzzy set of some underlying semantics.

For M being a set of few fuzzy or Boolean concepts, a large number of fuzzy concepts can be expressed by the elements of EM and the fuzzy logic operations can be implemented by the operations $\wedge$, $\vee$ and $'$ available in the EI algebra system (EM, $\wedge$, $\vee$, $'$) , even though we have not specified the membership functions of the fuzzy concepts in EM. In other words, the expressions and the fuzzy logic operations of the fuzzy concepts in EM just focus on the few simple concepts in M and the semantics of the fuzzy concepts in EM. As long as we can determine the fuzzy logic operations of these few concepts in M, the fuzzy logic operations of all concepts in EM can also be determined. Thus, not only will the accuracy of the representations and the fuzzy logic operations of fuzzy concepts be improved in comparison with the fuzzy logic equipped with some t-norms and a negation operator, but also the complexity of determining membership functions and their logic operations for the complex fuzzy concepts in EM will be alleviated. Let us stress that the complexity of human concepts is a direct result of the combinations of a few relatively simple concepts. It is obvious that the simpler the concepts in M, the more accurately and conveniently the membership functions and the fuzzy logic operations of the fuzzy concepts in EM will be determined. A collection of a few concepts in M plays a similar role to the one of a "*basis*" used in linear vector spaces. In what follows, we define the "*simple concepts*" which are suitable to serve as a "*basis*".

Definition 2. ([12]) Let ζ be any concept defined on the universe of discourse X. R_ζ is called a binary relation (i.e., $R_\zeta \subseteq X\times X$) of ζ if R_ζ satisfies: $x, y\in X$, $(x, y)\in R_\zeta \Leftrightarrow$ x belongs to concept ζ at some degree and the degree of x belonging to ζ is larger than or equal to that of y, or x belongs to concept ζ at some degree and y does not at all.

For instance, according to the value of each $x\in X$ on the feature of age shown in Table 1, we have the binary relations R_ζ, $R_{\zeta'}$, , R_γ of the fuzzy concepts ζ: "*old*", ζ': "*not old*", γ: "*the person whose age is about 40 years*" as follows

$$R_\zeta = \{(x, y) \mid (x, y)\in X\times X, age_x \geq age_y \},$$
$$R_{\zeta'} = \{(x, y) \mid (x, y)\in X\times X, age_x \leq age_y \},$$
$$R_\gamma = \{(x, y) \mid (x, y)\in X\times X, |age_x - 40| \leq |age_y - 40| \},$$

where age_x is the age of x. Note that $(x, x)\in R_\eta$ implies that x belongs to η to some degree and that $(x, x)\notin R_\eta$ implies that x does not belong to η at all. For the fuzzy concept m_5: "*more estate*" in Example 1, by feature "*estate*" and Definition 3, we have $(x_5, x_5)\in R_{m5}$ although the estate of x_5 is just 2, and $(x_2, x_2) \notin R_{m5}$ because the estate of x_2 is 0. For a Boolean concept ξ, $(x, x)\in R_\xi$ implies that x belongs to concept ξ. For instance, the concept m_6: "*male*" in Example 1, considering the feature "*male*" and Definition 2, we have (x_1, y), (x_4, y), (x_5, y), (x_7, y), $(x_8, y)\in R_{m6}$ and (x_2, y), (x_3, y), (x_6, y), (x_9, y), $(x_{10}, y)\notin R_{m6}$ for any $y\in X$.

In real world applications, the comparison of the degrees of a pair x and y belonging to a concept can be obtained through the use of the values of the feature or by relying on human intuition without representing such degrees by numbers in [0, 1] or considering a lattice. For instance, we can obtain the binary relation R_{m8} for fuzzy concept m_8: "*black hair persons*" in Example 1, just by comparing each pair of persons' hair and in this way articulating our intuitive judgment. Based on Table 1 and following this intuitive assessment, we can construct the binary relation R_m of each concept $m\in M$ being used in Example 1.

Definition 3. ([12]) Let X be a set and R be a binary relation on X. R is called a sub-preference relation on X if for $x, y, z \in X$, $x \neq y$, R satisfies the following conditions:

1. If $(x, y) \in R$, then $(x, x) \in R$;
2. If $(x, x) \in R$ and $(y, y) \notin R$, then $(x, y) \in R$;
3. If (x, y), $(y, z) \in R$, then $(x, z) \in R$;
4. If $(x, x) \in R$ and $(y, y) \in R$, then either $(x, y) \in R$ or $(y, x) \in R$.

A concept ζ is called a simple concept on X if R_ζ is a sub-preference relation on X. Otherwise ζ is called a complex concept on X. Here R_ζ is the binary relation of ζ defined by Definition 2.

Many concepts associated with more than a single feature (attribute) are complex concepts. For example, let X be a set of persons and cars. If $x, y \in X$, x is a person and y is a car and we consider concept "*beautiful*" , then the degrees of x, y belonging to "*beautiful*" are incomparable although both x and y may belong to "*beautiful*" at some degree, i.e., (x, x), $(y, y) \in R_{beautiful}$, $(y, x) \notin R_{beautiful}$, $(x, y) \notin R_{beautiful}$. This implies that 4 of Definition 3 is not satisfied and "*beautiful*" is a complex concept on X. By Table 1 and Definition 3, one can verify that each concept $m \in M$ in Example 1 is a simple concept. Let fuzzy concept $\beta = m_1 m_2$ meaning *"tall old persons"*. One can verify that (x_9, x_9), $(x_4, x_4) \in R_\beta$, but neither (x_9, x_4) nor (x_4, x_9) in R_β. This implies that the fourth condition of Definition 3 is not satisfied by R_β and therefore β is also a complex concept. The fuzzy concept $\gamma = \Sigma_{i \in I} (\prod_{m \in Ai} m) \in EM$ may be complex provided that $(x, y) \in R_\gamma \Leftrightarrow \exists k \in I$ such that $(x, y) \in R_{Ak}$ (i.e., $\forall m \in A_k$, $(x, y) \in R_m$) for $x, y \in X$. For instance, in Example 1, the fuzzy concept $\gamma = m_1 + m_2 \in EM$ reads as *"old persons"* or *"tall persons"*. By the data shown in Table 1, i.e., x_8: age=70, height=1.6; x_1: age=20, height=1.9; x_4: age=80, height=1.8, we have $(x_8, x_1) \in R_\gamma$ because x_8 is older than x_1 and $(x_1, x_4) \in R_\gamma$ because x_1 is taller than x_4. But x_8 is neither older nor taller than x_4, i.e., $(x_8, x_4) \notin R_\gamma$. Thus the binary relation R_γ does not satisfy 3 of Definition 3 and concept γ is a complex concept.

Definition 4. ([7]) Let $X_1, \ldots, X_n$, M be $n+1$ non-empty sets. Then the set $EX_1 \ldots X_n M^*$ is defined by

$$EX_1 \ldots X_n M^* = \{ \sum_{i \in I} (u_{1i} \ldots u_{ni} A_i) \mid A_i \subseteq M, u_{ri} \subseteq X_r, r = 1, \ldots, n, i \in I, I \text{ is a non-empty indexing set } \}.$$

In the case $n=0$,

$$EM^* = \{ \sum_{i \in I} A_i \mid A_i \subseteq M, i \in I, I \text{ is a non-empty indexing set } \}.$$

where the element $\sum_{i \in I} (u_{1i} \ldots u_{ni} A_i)$ is composed of terms $(u_{1i} \ldots u_{ni} A_i)$'s, $i \in I$, separated by "+". $\sum_{i \in I} (u_{1i} \ldots u_{ni} A_i)$ and $\sum_{i \in I} (u_{1p(i)} \ldots u_{np(i)} A_i)$ are the same element of $EX_1 \ldots X_n M^*$ if p is a bijection from I to I. When I is finite, $\sum^q_{i \in I} (u_{1i} \ldots u_{ni} A_i)$ is also denoted as $(u_{11} \ldots u_{n1} A_1) + \ldots + (u_{1q} \ldots u_{nq} A_q)$.

For a set, we know that the subsets of the set often contain or represent some useful information and knowledge. In real world applications, instead of one set, often many sets are involved and the information and knowledge represented by the subsets of different sets may have some kinds of relations. In order to study the complicated relations among the information and knowledge associated to different sets, we introduce the notation of $EX_1 \ldots X_n M^*$. Every element of $EX_1 \ldots X_n M^*$ is a "formal sum" of the terms constituted by the subsets of $X_1, \ldots, X_n$, M^*. For $\gamma = \sum_{i \in I} (u_{1i} \ldots u_{ni} A_i) \in EX_1 \ldots X_n M^*$, γ can be regarded as the "synthesis" of the information represented by all terms $u_{1i} \ldots u_{ni} A_i$'s. In practice, M is a set of elementary concepts, and $X_1, \ldots, X_n$,

are the sets associated the concepts in M. For example, let X be a set of persons and M be a set of concepts such as "*male*" , "*female*" , "*old*" , " *tall*" , "*high salary*" , " *black hair persons* " , "*white hair persons*" ,..., etc. For $\sum_{i\in I}(u_iA_i)\in EXM^*$, every term u_iA_i, $i\in I$, may mean that the persons in set $u_i\subseteq X$ satisfy some "condition" described by the concepts in $A_i\subseteq M$. AFS theory supports the studies on how to convert the information represented by the elements of $EX_1\ldots X_nM^*$ for the training examples and databases into the membership functions and their fuzzy logic operations.

Definition 5. ([7]) Let X_1, …, X_n, M be n+1 non-empty sets. A binary relation R on $EX_1\ldots,X_nM^*$ is defined as follows. For $\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)$, $\sum_{j\in J}(v_{1j}\ldots v_{nj}B_j)\in EX_1\ldots X_nM^*$,
$[\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)]R[\sum_{j\in J}(v_{1j}\ldots v_{nj}B_j)]\Leftrightarrow$

(i) $\forall(u_{1i}\ldots u_{ni}A_i)$ $(i\in I)$, $\exists(v_{1h}\ldots v_{nh}B_h)$ $(h\in J)$ such that $A_i\supseteq B_h$, $u_{ri}\subseteq v_{rh}$, $1\le r\le n$;

(ii) $\forall(v_{1j}\ldots v_{nj}B_j)$ $(j\in J)$, $\exists(u_{1k}\ldots u_{nk}A_k)$ $(k\in I)$ such that $B_j\supseteq A_k$, $v_{ri}\subseteq u_{rk}$, $1\le r\le n$;

It's obvious that R is an equivalence relation. The quotient set $EX_1\ldots X_nM^*/R$ is denoted by $EX_1\ldots X_nM$. By using notation $\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)=\sum_{j\in J}(v_{1j}\ldots v_{nj}B_j)$ we mean that $\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)$ and $\sum_{j\in J}(v_{1j}\ldots v_{nj}B_j)$ are equivalent under equivalence relation R and the membership degrees or the semantic meanings represented. In Example 1, $\xi=m_3m_8+m_1m_4+m_1m_6m_7+m_1m_4m_8$, $\zeta=m_3m_8+m_1m_4+m_1m_6m_7\in EM$, in the case n=0 for Definition 1, by Definition 2 we have $\xi=\zeta$.

Proposition 1. ([7]) Let X_1, …, X_n, M be n+1 non-empty sets. If $A_t\subseteq A_s$, $u_{rt}\supseteq u_{rs}$, r=1, 2, …., n, t, $s\in I$, $t\ne s$, $\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)\in EX_1\ldots X_nM$, then

$$\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)=\sum_{i\in I-\{s\}}(u_{1i}\ldots u_{ni}A_i) \tag{6}$$

Theorem 2. ([7]) Let X_1, …, X_n, M be n+1 non-empty sets. Then $(EX_1\ldots X_nM, \vee, \wedge)$ forms a completely distributive lattice under the binary compositions $\vee$ and $\wedge$ defined as follows. For any $\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)$, $\sum_{j\in J}(v_{1j}\ldots v_{nj}B_j)\in EX_1\ldots X_nM$,

$$\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)\vee\sum_{j\in J}(v_{1j}\ldots v_{nj}B_j)=\sum_{k\in I\sqcup J}(w_{1k}\ldots w_{nk}C_k), \tag{7}$$

$$\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)\wedge\sum_{j\in J}(v_{1j}\ldots v_{nj}B_j)=\sum_{i\in I,\ j\in J}[(u_{1i}\cap v_{1j}\ldots u_{ni}\cap v_{nj})(A_i\cup B_j) \tag{8}$$

where for any $k\in I\sqcup J$ (the disjoint union of I and J, i.e., every element in I and every element in J are always regarded as different elements in $I\sqcup J$), $C_k=A_k$, $w_{ik}=u_{rk}$ if $k\in I$, and $C_k=B_k$, $w_{ik}=v_{rk}$ if $k\in J$, r=1, 2, …, n.

$(EX_1\ldots X_nM, \vee, \wedge)$ is called the EI^{n+1} (expanding n+1 sets X_1, …, X_n, M) algebra over X_1, …, X_n and M, one kind of AFS algebra. For $\alpha=\sum_{i\in I}(u_{1i}\ldots u_{ni}A_i)$, $\beta=\sum_{j\in J}(v_{1j}\ldots v_{nj}B_j)\in EX_1\ldots X_nM$, $\alpha\le\beta\Leftrightarrow\alpha\vee\beta=\beta\Leftrightarrow\forall(u_{1i}\ldots u_{ni}A_i)$ $(i\in I)$, $\exists(v_{1h}\ldots v_{nh}B_h)$ $(h\in J)$ such that $A_i\supseteq B_h$, $u_{ri}\subseteq v_{rh}$, $1\le r\le n$.

Matlab application methods 4:

In the case n=1, *the elements of EXM* are represented by matrices. For* $\sum_{i\in I}(u_{1i}A_i)\in EXM^*$, *the term* $u_{1i}A_i$ *is represented by the i*th *column of the matrix*, $i\in I$. *The upper part and the lower part of column i which are separated by zero represent the set* A_i, u_i *respectively. For example*, $M=\{m_1, m_2, \ldots, m_{12}\}$ *in Example* 1.*For* $\eta=\{x_1x_3x_4\}$ $\{m_1m_2m_4m_7\}$, $\zeta=\{x_1x_3x_4x_5\}$ $\{m_1m_4\}$ $+\{x_7x_9\}$ $\{m_2m_5m_6\}\in EXM^*$. U= [0 1 2 4 7 0 1 3 4]', V= [0 0 1 4 0 1 3 4 5; 0 2 5 6 0 0 0 7 9]', *where matrices U*, *V represent the EII elements* η, ζ. *About EII algebra*, *we always employ functions*: reduceEII(*M*), EIIsum(*M*,*N*), EIImult(*M*,*N*) *to reduce an EII element in EXM*or execute the binary compositions* $\vee$ *and* $\wedge$ *on EII elements represented by matrices M*, *N*.

2. *AFS structure of data*

An AFS structure, a triple (M, τ, X), gives rise to various lattice representations of the membership degrees and fuzzy logic operations of the concepts in *EM* [11].

Definition 6. ([7], [9]) Let X, M be sets and 2^M be the power set of M. Let $\tau: X\times X\to 2^M$. (M, τ, X) is called an AFS structure if τ satisfies the following axioms:

AX1: $\forall(x_1, x_2)\in X\times X,\ \tau(x_1, x_2)\subseteq \tau(x_1, x_1)$;

AX2: $\forall(x_1, x_2), (x_2, x_3)\in X\times X,\ \tau(x_1, x_2)\cap\tau(x_2, x_3)\subseteq \tau(x_1, x_3)$.

X is called universe of discourse; M is called a concept set and τ is called a structure.

Let X be a set of objects and M be a set of simple concepts on X. If $\tau: X\times X\to 2^M$ is defined as follows: for any $(x, y)\in X\times X$

$$\tau(x,\ y)=\{m \mid m\in M,\ (x,\ y)\in R_m\}\in 2^M \tag{4}$$

where R_m is the binary relation of simple concept $m\in M$ (refer to Definition 2). Then (M, τ, X) is an AFS structure. Now we prove this. For any $(x_1, x_2)\in X\times X$, if $m\in\tau(x_1, x_2)$, then by (4) we know that $(x_1, x_2)\in R_m$. Because each $m\in M$ is a simple concept, we have $(x_1, x_1)\in R_m$ by Definition 3, i.e., $m\in\tau(x_1, x_1)$. This implies that $\tau(x_1, x_2)\subseteq\tau(x_1, x_1)$ and AX1 of Definition 4 holds. For (x_1, x_2), $(x_2, x_3)\in X\times X$, if $m\in\tau(x_1, x_2)\cap\tau(x_2, x_3)$, then (x_1, x_2), $(x_2, x_3)\in R_m$. Since m is a simple concept, so by Definition 3, we have $(x_1, x_3)\in R_m$, i.e., $m\in\tau(x_1, x_3)$. This implies $\tau(x_1, x_2)\cap\tau(x_2, x_3)\subseteq \tau(x_1, x_3)$ and AX2 of Definition 4 holds. Therefore (M, τ, X) is an AFS structure. By the above discussion, an AFS structure based on a data set can be established by (4), as long as each concept in M is a simple concept on X.

Let us continue with Example 1, in which $X=\{x_1, x_2, \ldots, x_{10}\}$ is the set of 10 persons and their feature descriptions are shown in Table 1. $M=\{m_1, m_2, \ldots, m_{12}\}$ is the set of simple concepts shown in Example 1. By Table 1 and Definition 3, one can verify that each concept $m\in M$ is a simple concept. Thus for any $x, y\in X$, $\tau(x, y)$ is well-defined by (4). For instance, we have

$$\tau(x_4, x_4)=\{m_1, m_2, m_3, m_4, m_5, m_6, m_8, m_9, m_{10}, m_{11}, m_{12}\}$$
$$\tau(x_4, x_7)=\{m_1, m_2, m_6, m_9, m_{10}\}$$

by comparing the attribute values of x_4, x_7 shown in Table 1 as follows:

	age	*height*	*weight*	*Salary*	*estate*	*male*	*female*	*black*	*white*	*yellow*
x_4	80	1.8	73	20	80	1	0	3	4	2
x_7	45	1.7	78	268	90	1	0	1	6	4

Similarly, we can obtain $\tau(x, y)$ for other $x, y\in X$. Finally we have the AFS structure (M, τ, X) corresponding to Table 1.

Matlab application methods 5:

AFS structures are represented by 3-dimension Boolean matrices. In Example 1, *AFS structure* (M, τ, X) *is represented by 3-dimension Boolean matrix named Str_Mat, in which the Boolean vector Str_Mat*$(j, :, i)=(b_1, b_2, \ldots, b_{12})$ *represents the set* $\tau(x_i, x_j)$ *such that* $m_k\in\tau(x_i, x_j)\Leftrightarrow b_k=1$. *For example, Str_Mat*(4,:,4)=[1 1 1 1 1 1 1 1 1 1 1 1]', *Str_Mat*(7,:,4)=[1 1 0 0 0 1 0 0 1 1 0 0]'.

We employ the function: gen_structure(*Data_Mat*, *Parameter_Mat*) *to generate an AFS structure* (M, τ, X) *by the semantic meanings with the parameters in Parameter_Mat and the original data in Data_Mat*. *The semantic meanings of the simple concepts in* M *are determined by the parameters in Parameter_Mat shown as the following example. In Example* 1, Data_Mat(*i*, *j*) *is the number in the* i*th row* and *j*th *column of Table* 1. *Parameter_Mat*=[80 40 3; *NaN NaN* 1.9; *NaN NaN* 90; *NaN NaN* 268; *NaN NaN* 98; *NaN NaN* 1; *NaN NaN* 1; *NaN NaN* 1; *NaN NaN* 1; *NaN NaN* 1]'. *Then the function* gen_structure(*Data_Mat*, *Parameter_mat*) *returns an AFS structure* (M, τ, X) *represented by a 3-dimension Boolean matrix*, *in which* M={m_1, m_2, ….m_{24}}, m_1: "*old persons*", m_2= m_1', i.e., m_2: "*not old persons*"; m_3: "*about 40 years old persons*", m_4= m_3'; m_5: "*young persons*", m_6= m_5'; m_7: "*tall persons*", m_8= m_7'; m_9: "*heavy persons*", m_{10}= m_9'; m_{11}: "*high salary*", m_{12}= m_{11}'; m_{13}: "*more eatate*", m_{14}= m_{13}'; m_{15}: "*male*", m_{16}= m_{15}'; m_{17}: "*female*", m_{18}= m_{17}'; m_{19}: "*black hair persons*", m_{20}= m_{19}'; m_{21}: "*white hair persons*", m_{22}= m_{21}'; m_{23}: "*yellow hair persons*", m_{24}= m_{23}'.

Notice: *The labels of the simple concepts in* M *are determined by matrix Parameter_Mat as the following rules*: *matrix* [1 3 5; *NaN NaN* 7; *NaN NaN* 9; *NaN NaN* 11; *NaN NaN* 13; *NaN NaN* 15; *NaN NaN* 17; *NaN NaN* 19; *NaN NaN* 21; *NaN NaN* 23]' *represents the set of the labels of the simple concepts in* M.

3. *The representations of fuzzy concepts in EM*

In this section, we recall some representations of fuzzy concepts in the framework of AFS theory which will be used in the sequel.

Theorem 2. ([7]) Let (M, τ, X) be an AFS structure. For $x\in X$, $A\subseteq M$, we define the symbol

$$A^{\tau}(x)=\{\ y\mid y\in X,\ \tau(x,\ y)\supseteq A\ \}. \tag{11}$$

For any given $x\in X$, if we define ϕ_x: $EM\to EXM$ as follows: For any $\sum_{i\in I}(\prod_{m\in Ai}m)\in EM$,

$$\phi_x(\sum_{i\in I}(\prod_{m\in Ai}m))=\Sigma_{i\in I}\ A_i^{\tau}(x)A_i\in EXM,$$

then ϕ_x is a homomorphism from the lattice (EM, $\wedge$, $\vee$) to the lattice (EXM, $\wedge$, $\vee$).

Matlab application methods 6:

We employ function under_A_xi(*Str_Mat, Set_Simple_Concept, Index_Xi*) *to find set* $A^{\tau}(x_i)$ *in AFS structure* (M, τ, X), *where the 3-dimension Boolean matrix Str_Mat represents the AFS structure* (M, τ, X). *The column vector Set_Simple_Concept represents the labels of the simple concepts in set* $A\subseteq M$. *Index_Xi is the label of* x_i.

Theorem 2 implies that for any given concept $\sum_{i\in I}(\prod_{m\in Ai}m)\in EM$, we get a map $\sum_{i\in I}(\prod_{m\in Ai}m)$: $X\to EXM$, $\forall x\in X$,

$$(\sum_{i\in I}(\prod_{m\in Ai}m))(x)=\Sigma_{i\in I}A_i^{\tau}(x)A_i\in EXM. \tag{12}$$

Since (EXM, $\wedge$, $\vee$) is a lattice, hence map $\sum_{i\in I}(\prod_{m\in Ai}m)$ is a L-fuzzy set with membership degrees valued by EII algebra EXM. For α, $\beta\in EM$, $\alpha\vee\beta$ and $\alpha\wedge\beta$ are "*or*" and "*and*" of the L-fuzzy sets α and β respectively. " ′ " is the negation of the L-fuzzy sets in EM. Thus we have an EII algebra representations and logic operations of fuzzy concepts in EM, and (EM, $\wedge$, $\vee$, ′) can be viewed as a fuzzy logic system.

Although EII algebra representations of fuzzy concepts preserve more original information

than $E^{\#}I$ algebra representations in [11], there may be still too many elements of X whose degrees belonging to some fuzzy concepts in EM can not be compared by EII algebra values. In some cases, it is not expedient for us to apply EII algebra representations to solve the real world problems. Thus we proposed the $E^{\#}I^n$ algebras in [11] and gave its representations of fuzzy concepts in EM. In the following, we recall what will be used in this paper.

Definition 7. ([11]) Let X be a non-empty set. A binary relation $R^{\#}$ on the set

$$EX^*=\{\ \Sigma_{i\in I}a_i \mid a_i\in 2^X,\ I \text{ is a non-empty indexing set }\}$$

is defined as follows. For $\Sigma_{i\in I}a_i$, $\Sigma_{j\in J}b_j\in EX^*$, $(\Sigma_{i\in I}a_i)R^{\#}(\Sigma_{j\in J}b_j)\Leftrightarrow$ i) $\forall a_i$ $(i\in I)$, $\exists b_h$ $(h\in J)$ such that $a_i\subseteq b_h$. ii) $\forall b_j$ $(j\in J)$, $\exists a_k$ $(k\in I)$ such that $b_j\subseteq a_k$.

It is obvious that $R^{\#}$ is an equivalence relation on EX^*. The quotient set $EX^*/R^{\#}$ is denoted by $E^{\#}X$. By using notation $\Sigma_{i\in I}a_i=\Sigma_{j\in J}b_j$ we mean that $\Sigma_{i\in I}a_i$ and $\Sigma_{j\in J}b_j$ are equivalent under the equivalence relation $R^{\#}$ and the membership degrees represented by them are equal.

Proposition 2. ([11]) Let X be a non-empty set. For $\Sigma_{i\in I}a_i\in E^{\#}X$, if $a_u\subseteq a_v$, $v\in I$, $u\neq v$, then $\Sigma_{i\in I}a_i=\Sigma_{i\in I-\{u\}}a_i$.

Theorem 3. ([11]) Let X be a non-empty set. Then $(E^{\#}X, \wedge, \vee)$ forms a completely distributive lattice under the binary compositions $\wedge$, $\vee$ defined as follows. For any $\Sigma_{i\in I}a_i$, $\Sigma_{j\in J}b_j\in E^{\#}X$,

$$(\Sigma_{i\in I}a_i)\vee(\Sigma_{j\in J}b_j)=\Sigma_{k\in I\bigsqcup J}c_k, \tag{13}$$

$$(\Sigma_{i\in I}a_i)\wedge(\Sigma_{j\in J}b_j)=\Sigma_{i\in I,j\in J}(a_i\cap b_j), \tag{14}$$

where for any $k\in I\bigsqcup J$ (the disjoint union of I and J), $c_k=a_k$ if $k\in I$, and $c_k=b_k$ if $k\in J$.

$(E^{\#}X, \wedge, \vee)$ is called an $E^{\#}I$ algebra over X. $\varnothing$, X are the minimum and maximum element in $E^{\#}X$ respectively. $E^{\#}I$ algebra representation of each fuzzy concept in EM is obtained by Proposition 3.

Matlab application methods 7:

The elements of $E^{\#}X$ are also represented by matrices. For example, $X=\{x_1, x_2, \ldots, x_{10}\}$ in Example 1. *$\eta=x_1x_2x_3x_9$, $\zeta=x_1x_3x_5x_7x_9+x_2x_5x_6\in E^{\#}X$. U=[0 1 2 3 9]', V=[0 1 3 5 7 9; 0 0 0 2 5 6]', where matrices U, V represent the $E^{\#}I$ elements η, ζ. About $E^{\#}I$ algebra, we always employ functions*: reduceEIco(*M*), EIcosum(*M*,*N*), EIcomult(*M*,*N*), equal_EIco(*M*,*N*), less_EIco(*M*,*N*) *to reduce an $E^{\#}I$ element in $E^{\#}X$, execute the binary compositions $\vee$ and $\wedge$ on $E^{\#}X$ elements represented by matrices M and N, and judge the equivalence or the size relations of $E^{\#}X$ elements*.

Proposition 3. ([11]) Let X, M be two non-empty sets, and let EXM and $E^{\#}X$ be EI^2 and $E^{\#}I$ algebra respectively. If for any $\Sigma_{i\in I}(u_iA_i)\in EXM$, map p is defined by $p[\Sigma_{i\in I}(u_iA_i)]=\Sigma_{i\in I}u_i$, then p is a homomorphism from the lattice $(EXM, \wedge, \vee)$ to the lattice $(E^{\#}X, \wedge, \vee)$.

Given an AFS structure (M,τ, X) and for each $x\in X$, we have two homeomorphisms ϕ_x (defined by Theorem 2): $EM\rightarrow EXM$, p (defined by Proposition 3) $EXM\rightarrow E^{\#}X$. The composed

map $p\cdot\phi_x$: $EM\rightarrow E^{\#}X$ is also a homomorphism from the lattice (EM, $\wedge$, $\vee$) to the lattice ($E^{\#}X$, $\wedge$, $\vee$). For each fuzzy concept $\alpha=\sum_{i\in I}(\prod_{m\in Ai}m)\in EM$, we get another kind of L-fuzzy set representations of fuzzy concept α, i.e., the $E^{\#}I$ algebra representations as follows. For any $x\in X$

$$\alpha(x)=p\cdot\phi_x(\alpha)=\Sigma_{i\in I}A_i^{\tau}(x)\in E^{\#}X. \tag{15}$$

In [11], the authors have proposed a special family of measures using which the *EII*, $E^{\#}I$ algebras become a lattice with norms. Such that we can convert the *EII*, $E^{\#}I$ algebra represented membership degrees into [0,1] interval and the information contained in the algebra representations can be also preserved to a significant extent.

Definition 8 ([12]) Let γ be a simple concept on X, ρ_γ: $X \rightarrow R^+ = [0, \infty)$. ρ_γ is called a weight function of simple concept γ if ρ_γ satisfies the following conditions:

1. $\rho_\gamma(x) = 0$, $(x, x)\notin R_\gamma$, $x\in X$;
2. $\rho_\gamma(x)\geq \rho_\gamma(y) \Leftrightarrow (x, y)\in R_\gamma$, $x, y\in X$,

where R_γ is the binary relation of concept γ (refer to Definition 2).

For example, if $X\subseteq R^n$ and m_{i1}, m_{i2}, m_{i3}, $m_{i4}\in M$; i = 1, 2, …, n, are the fuzzy concepts, "*small*" , "*medium*" , "*no-medium*", "*large*" associating to the feature f_i respectively, the weight functions of them can be defined according to the observed data X and their semantic interpretations as follows:

$$\rho_{m_{j1}}(x_i)=\frac{h_{j1}-f_j(x_i)}{h_{j1}-h_{j2}} \tag{16}$$

$$\rho_{m_{j2}}(x_i)=\frac{h_{j4}-|\,f_j(x_i)-h_{j3}\,|}{h_{j4}-h_{j5}} \tag{17}$$

$$\rho_{m_{j3}}(x_i)=\frac{|\,f_j(x_i)-h_{j3}\,|-h_{j5}}{h_{j4}-h_{j5}} \tag{18}$$

$$\rho_{m_{j4}}(x_i)=\frac{f_j(x_i)-h_{j2}}{h_{j1}-h_{j2}} \tag{19}$$

where j = 1, 2, …, n;

h_{j1} =max{$f_j(x_1)$, $f_j(x_2)$, … , $f_j(x_h)$};
h_{j2} =min{$f_j(x_1)$, $f_j(x_2)$, …, $f_j(x_h)$};
h_{j3} =($f_j(x_1)$ + $f_j(x_2)$ + … + $f_j(x_h)$)/h;
h_{j4} =max{ $|f_j(x_k) - h_{j3}|$ $\mid k = 1, 2, \ldots, h$};
h_{j5} =min{ $|f_j(x_k) - h_{j3}|$ $\mid k = 1, 2, \ldots, h$}.

By Definition 8 and the semantic interpretations of m_{i1}, m_{i2}, m_{i3}, m_{i4} one can verifies that ρ_{mjk}, j = 1, 2,…, n are weight functions of simple concept m_{jk}. In general, the weight function ρ_γ of a simple concept γ associating to a feature f_i is subjectively defined by users according to the data distribution on the feature f_i and the semantic interpretation of the simple concept γ. It is obvious that for a given simple concept γ we can define many different functions ρ_γ: $X \rightarrow[0, \infty)$ such that satisfies the weight function conditions shown in Definition 8. The diversities of the weight functions of a simple concept result from the subjective imprecision of human perception of the

observed data. However, the diversities are bounded or constrained by the sub-preference relation of the simple concepts defined by Definition 3 according to the semantics of the natural language. This is rooted in the fact that perceptions are intrinsically imprecise, reflecting the bounded ability of sensory organs. In order to provide a tool kit for representing and managing an infinitely complex reality, the weight functions for simple concepts are mental constructs with the subjective imprecision (i.e., subjectively constructing the functions satisfying Definition 8 for the concerned simple concepts). But the constructs of the weight function for a simple concept γ have to observe the objectivity in nature, which is the sub-preference relation R_γ (refer to Definition 3) objectively determined by the observed data X and the semantics of γ. In other words, the subjective imprecision of the weight function of γ are constrained by the objectivity of R_γ. The multi-options of the weight functions just reflect the subjective imprecision of the perceptions of the observed data.

Definition 9. ([11]) Continuous case: Let X be a set, $X\subseteq R^n$ and S ($S\subseteq 2^X$) be the set of Borel sets in X. $\rho : X\to[0, \infty)$ is integrable on X under Lebesgue measure with $0<\int_X \rho d\mu<\infty$. For all $A\in S$, we define a measure m_ρ over S as follows:

$$m_\rho(A)=\frac{\int_A \rho d\mu}{\int_X \rho d\mu}. \tag{16}$$

Discrete case: Let X be a set and $S\subseteq 2^X$ be a σ-algebra over X. $\rho : X\to[0, \infty)$ with $0<\Sigma_{x\in X}\rho(x)<\infty$. For any $A\in S$, a measure m_ρ over σ-algebra S is defined as follows:

$$m_\rho(A)=\frac{\sum_{x\in A}\rho(x)}{\sum_{x\in X}\rho(x)}. \tag{17}$$

Given [4], we can verify that m_ρ defined in Definition 8 is a measure over X for functionρ. $\rho(x)$ may have various explanations in different situations. In general, $\rho(x)$ weights the essentiality of the sample x for the category of concepts under consideration. For example, if we want to find the description of old persons according to the data like Table 1, the weight of the samples of old persons should be higher than the samples of young persons. We employ the weight function $\rho(x)$ to reduce the influence of less essential samples and increase the influence of more essential samples when we determine the membership functions by the measure.

Let's consider the $E^{\#}I$ algebra valued membership degrees defined by (15). Let $S\subseteq 2^X$ be an σ-algebra over X and m be a measure on S with $0\le m(A)\le 1$ for any $A\in S$. For the fuzzy concept $\eta=\sum_{i\in I}(\prod_{m\in Ai} m) \in EM$, if $\forall\ x\in X$, $\forall i\in I$, $A_i^{\tau}(x) \in S$, then the membership function of η is defined as follows:

$$\mu_\eta(x)= \sup_{i\in I}\{m(A_i^{\tau}(x))\}\in[0, 1]. \tag{18}$$

for any $x\in X$. In general, the measure m is constructed according to the group of concepts under consideration. In practice, the measure m can be induced by a function $\rho : X\to[0, \infty)$ as will be clarified in the forthcoming definition (Definition 9).

Let us further study the data in Example 1. Let (M, τ, X) be the AFS structure established in

this example and $S=2^X$ be the σ-algebra over X. Let $\alpha=m_2m_3m_5+m_2m_6m_7+m_2m_9+m_2m_3m_4+m_4m_5m_6m_7+m_4m_5m_9+m_3m_4m_5+m_6m_8m_9$, $\beta=m_6m_8m_9+m_4m_5+m_2$. By Theorem 1, we can verify that $\alpha\leq\beta$ in *EI* algebra *EM* and for the membership functions of fuzzy concept α and β defined by (18) with any measure m, we always have $\mu_\alpha(x)\leq\mu_\beta(x)$. Suppose that each sample in X is equally relevant to the concepts in M we have considered. Thus the measure "m" could be defined as the ratio expressed as $\forall W\in S$, $m(W)=|W|/|X|$, where $|W|$ is the number of the elements in W. Let $\nu=m_1m_7+m_2m_7$, $\gamma=m_7m_9+m_5m_7m_{10}\in EM$. Table 2 shows the membership degrees of each person belonging to the fuzzy concepts with the semantics captured by $m_1m_7+m_2m_7$ (i.e., "*old female*" or "*tall female*") and $m_7m_9+m_5m_7m_{10}$ (i.e., "*white hair female*" or "*more estate yellow hair female*").

Table 2-The membership functions defined by (18)

fuzzy concepts	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
$\mu_\nu(.)$	0	0.2	0.7	0	0	0.5	0	0	0.9	0.1
$\mu_\gamma(.)$	0	0.6	1.0	0	0	1.0	0	0	1.0	0.2
age	20	13	50	80	34	37	45	70	60	3
height	1.9	1.2	1.7	1.8	1.4	1.6	1.7	1.65	1.82	1.1
estate	0	0	34	80	2	28	90	45	98	0
female	0	1	1	0	0	1	0	0	1	1
white hair	1	3	1	4	2	1	6	4	3	5
yellow hair	4	1	4	2	2	4	4	2	1	3

In Table 2, the degrees of membership that persons x_3 and x_6 belong to γ are 1. Their feature descriptions are female and number 1 for the order of hair color which coincides with the semantic "*white hair female*" to the highest extent. Because both ν and γ associate with feature female in their semantics, the membership degrees of all male persons in X belonging to both ν and γ are 0. For x_9 who is characterized by the height of 1.82 and who is female, we can conclude that she belongs to the fuzzy concept "*tall female*" at 0.9. Similarly, one can observe that other membership degrees shown in Table 2 are consistent with our intuitive assessment of the persons described by the data. In general, $\mu_\eta(x)$ defined by (18) is the membership degree of x belonging to the fuzzy concept η with the semantics expressed by $\sum_{i\in I}(\prod_{m\in Ai}m)$. $\mu_\eta(x)$ is determined by $A_i^\tau(x)$ and the measure m. Here $A_i^\tau(x)\subseteq X$ is the set of all samples in X whose degree belonging to concept $\prod_{m\in Ai}m$ are less than or equal to that of x and is determined by both the AFS structure (M, τ, X) and the semantics of the simple concepts in A_i. The measure m evaluates the total essentiality of the samples in $A_i^\tau(x)$ to give the degree of x belonging to concept $\prod_{m\in Ai}m$. The AFS structure (M, τ, X) is determined by the given data like Example 1 and the semantics of simple concept in M. The semantics of the fuzzy set with the membership function defined in (18) is expressed by $\sum_{i\in I}(\prod_{m\in Ai}m)$.

In what follows, we consider the *EII* algebra valued membership degrees defined by (12).

Definition 10 ([11]) Let X, M be non-empty sets. Let M be a finite set of simple concepts on X, *EXM* be EI^2 algebra over X, M, and S be a σ-algebra over X. For every simple concept $\zeta\in M$, m_ζ is the measure defined by Definition 9 for ρ_ζ the weight function of ζ defined by Definition 8. If $\delta=\sum_{i\in I}a_iA_i\in EXM$ satisfies $a_i\in S$, for any $i\in I$, then $\|\delta\|$, the norm of δ is defined as follows.

$$\| \sum_{i\in I} a_i A_i \| = \sup_{i\in I} (\prod_{\gamma\in A_i} m_\gamma(a_i)) \tag{19}$$

Or

$$\| \sum_{i\in I} a_i A_i \| = \sup_{i\in I} (\inf_{\gamma\in A_i} m_\gamma(a_i)) \tag{20}$$

For a_kA_k, $k\in I$, *if* $A_k=\varnothing$, we define

$$\prod_{\gamma\in A_k} m_\gamma(a_k) = \inf_{\gamma\in A_k} m_\gamma(a_k) = \max_{\gamma\in M} \{m_\gamma(a_k)\} \tag{21}$$

Let (M, τ, X) be an AFS structure and ||.|| be a norm defined by Definition 10. For any fuzzy concept $\eta = \sum_{i\in I}(\prod_{m\in Ai} m) \in EM$, if $\forall\ x\in X$, $\forall i\in I$, $A_i^\tau(x) \in S$, then the membership function of η, which is based on the *EII* algebra valued membership degrees defined by (12), is defined as follows. For any $x\in X$,

$$\mu_\eta(x) = \| \eta(x) \| = \| \sum_{i\in I} A_i^\tau(x) A_i \| \tag{22}$$

In the above discussion, we know that $A_i^\tau(x)$ is the set of all elements in X whose degree of belonging to concept $\prod_{m\in Ai}m$ is less than or equal to that of x. By (11), we have

$$A_i^\tau(x) = \bigcap_{\gamma\in A_i} \{\gamma\}^\tau(x)$$

By Example 1 and (11), we know that $\{\gamma\}^\tau(x)$ is determined by both the distribution of the original data and the semantic meaning of m. Thus $A_i^\tau(x)$ is dependent on both the distribution of the original data, i.e., randomness (objective uncertainty) and the semantic meanings of simple concepts $\gamma\in A_i$, i.e., fuzziness (subjective imprecision). In general, the larger the set $A_i^\tau(x)$, the larger the degree x belonging to $\prod_{m\in Ai} m$ if all elements in X have the equally essential relation to the considering group of concepts, i.e., for any x, $y\in X$, $\rho(x)=\rho(y)$. If some samples in X have so inappreciable relationship with the considering group of concepts that whether or not they are included in $A_i^\tau(x)$ has not significative influence on the evaluating the degree of x belongingness (membership) to $\prod_{m\in Ai} m$. Thus $\rho(x)$'s for these samples should be very small or 0. Since $A_i^\tau(x)$ and $A_i^\tau(y)$ are independent on $\rho(x)$ and $\rho(y)$, hence $\rho(x)>\rho(y)$ does not mean that $A_i^\tau(x)\supseteq A_i^\tau(y)$ or $m(A_i^\tau(x)) > m(A_i^\tau(y))$. In other words, the more essential a sample is does not entail that the degree of the membership to the fuzzy concept in *EM* is always higher. In other words, $\rho(.)$ weights the referring value of each sample in X for the determining of the membership functions of fuzzy concepts in *EM*. By measure theory [4], we know that the measurement of $\mu_\gamma(x) = m_\gamma(\{\gamma\}^\tau(x))$, which is the measurement of the degree of x belonging to simple concept $\gamma\in A_i$, is not enough to determined $\mu_{\prod_{m\in A_i} m}(x) = \prod_{\gamma\in A_i} m_\gamma(A_i^\tau(x))$. While in the fuzzy logic systems equipped by the t-norm, $\mu_{\prod_{m\in A_i} m}(x) = \mathop{T}_{\gamma\in A_i} (\mu_\gamma(x))$ is just determined by $\mu_\gamma(x)$, $\gamma\in A_i$ and independent on the distribution of the original data. In other words, the member-
ship functions defined by (18) or (22) taking both fuzziness (subjective imprecision) of the semantics of the simple concepts and randomness (objective uncertainty) of the distribution of the original data into account.

Matlab application methods 8:

We employ the function: degree_xi(*Lou_Mat*,*Str_Mat*, *Fuzzy_Set*, *Index_Xi*, *Logic_Ind*) *to obtain the membership degree of the sample labeled by Index_Xi belonging to the fuzzy concept* (*in EM*) *represent by matrix Fuzzy_Set in the AFS structure represented by the* 3*-demension Boolean matrix Str_Mat. The weight functions of the simple concepts in M are represented by Lou_Mat*, *for detail*, *Lou_Mat*(*i*, *j*)=$\rho_{mj}(x_i)$. *Logic_Ind* =1, *the function returns the membership degree defined by formula* (19); *Logic_Ind* =2, *the function returns the membership degree defined by formula* (20).

References

[1] L.A. Zadeh, Fuzzy sets, *Information and Control*, 8(1965)338-353.

[2] J. E. Graver, M. E. Watkins, Combinatorics with Emphasis on the Theory of Graphs. inc.:Springer-Verlag, New York, 1977.

[3] G. J. Wang, " Theory of topological molecular lattices, " *Fuzzy Sets and Systems*, vol. 47, pp. 351-376, 1992.

[4] P. R. Halmos, Measure theory, Springer-Verlag, New York. 1974.

[5] D. Kim, " Data classification based on tolerant rough set, " *Pattern Recognition,* vol. 34, pp.1613-1624, 2001.

[6] C. J. Merz, P. M. Murphy. (1996) UCI Repository for Machine Learning Data-Bases. Dept. of Information and Computer Science, University of California, , Irvine, CA. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[7] X. D. Liu, " The Fuzzy Theory Based on AFS Algebras and AFS Structure," *Journal of Mathematical Analysis and Applications*, vol. 217, pp. 459-478, 1998.

[8] X. D. Liu, " The Topology on AFS Algebra and AFS Structure," *Journal of Mathematical Analysis and Applications*, vol. 217, pp. 479-489, 1998.

[9] X. D. Liu, " The Fuzzy Sets and Systems Based on AFS Structure, EI Algebra and EII algebra," *Fuzzy Sets and Systems*, vol. 95, pp. 179-188, 1998.

[10] X D. Liu, W. Pedrycz and Q. L. Zhang, " Axiomatics Fuzzy sets logic," *The Proceedings of 2003 IEEE International Conference on Fuzzy Systems*, vol. 1, pp.55-60, 2003.

[11] X. D. Liu, T. Y. Chai and W. Wang, " Approaches to the Representations and Logic Operations for Fuzzy Concepts in the Framework of Axiomatic Fuzzy Set Theory I, II, " *Information Sciences*, 177 (2007) 1007–1026, 1027–1045.

[12] X. D. Liu, W. Wang, T. Y. Chai, " The Fuzzy Clustering Analysis Based on AFS Theory," *IEEE Transactions on Systems, Man and Cybernetics* Part B, vol. 35, no.5, pp.1013-1027, 2005.

[13] X. D. Liu, W. Pedrycz, " The Development of Fuzzy Decision Trees in the Framework of Axiomatic Fuzzy Set Logic, " *Applied Soft Computing*, 7(2007)325-342.

[14] X. D. Liu, The Structure of Fuzzy Matrices, *Journal of Fuzzy Mathematics*, 2(1994)311-325. 52

[15] X. D. Liu , A New Mathematical Axiomatic System of Fuzzy Sets and Systems, *Journal of Fuzzy Mathematics*, 3(1995)559-560.

[16] X. D. Liu, A new fuzzy model of pattern recognition and hitch diagnoses of complex systems, *Fuzzy Sets and Systems*, 104(1999)289-297.

[17] X D. Liu, W. Pedrycz and Q. L. Zhang, Axiomatics Fuzzy sets logic, *The Proceedings of IEEE International Conference on Fuzzy Systems*, 2003,1:55-60.

[18] X. D. Liu, Zhu Kejiu, Huang Hongzhong,The Representations of Fuzzy Concepts Based on the Fuzzy Matrix Theory and the AFS Theory, *IEEE International Symposium on Intelligent Control*, October 5-8, pp.1006-1011, Houston, Texas, USA, 2003.

[19] X. D. Liu, Zhang Q. L., The Fuzzy Cognitive Maps Based on AFS Fuzzy Logic, *Dynamics of Continuous, Discrete and Impulsive Systems*, Series A: Mathematical Analysis, 11(5-6)(2004)787-796.

[20] X. D. Liu, Wanquan Liu, Credit Rating Analysis with AFS Fuzzy Logic, *Lecture Notes in Computer Science*, LNCS 3612(2005)1198-1204.

[21] X. D. Liu, AFS Theory and Its Applications, *IEEE Systems, Man and Cybernetics, Society eNewsletter*, 13:issue, December 2005, available on-line: http://www.ieeesmc.org/Newsletter/Dec2005/R8xdliu.php.

[22] X. D. Liu, T. Y. Chai and W. Wang, AFS Fuzzy Logic Systems and Its Applications to Model and Control, *International Journal of Information and Systems Sciences*, 2(3)(2006)285-305.

[23] X. D. Liu, L. S. Zhang, J Zhou, K. J. Zhou and Q. L. Zhang, The Structures of EI Algebras Generated by Information Attributes, *Int. J. Intelligent Systems Technologies and Applications*, in press, 2006.

[24] Y. Ren, M. Song, and X. Liu, New approaches to the fuzzy clustering via AFS theory, *International Journal of Information and Systems Sciences*, 3(2)(2007)307-325.

[25] L. S. Zhang and X. D. Liu, Concept Lattice and AFS Algebra, *Lecture Notes in Computer Science*, LNAI 4223(2006)290-299.

[26] Y. J. Zhang, D. Q. Liang and S. C. Tong, On AFS Algebra Part I, *Information Sciences*, 167(2004)263-286.

[27] Y. J. Zhang, D. Q. Liang and S. C. Tong, On AFS Algebra Part II, *Information Sciences*, 167(2004)287-303.

[28] R. Ding, X. D. Liu and Y. Chen, The Fuzzy Clustering Algorithm based on AFS Topology, *Lecture Notes in Computer Science*, LNAI 4223(2006)89-98.

Appendix: *The Original Matlab Codes*

1. compactEI

Compact an *EI* element.

Syntax

N=compactEI(*M*)

Description

N=compactEI(*M*) returns a matrix *N*, which represents an *EI* element in *EM* equal to what is represented by *M*. The numbers of rows of *N* are the least of all matrices which represent the same *EI* element as *M*. Each column of *N* is sorted in ascending order.

Examples

M=[0,1,2,3,3,4; 0,0,3,2,5,4; 0,1,1,6,7,4; 0,0,0,2,4,5]’

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2

3 5 7 4
4 4 4 5

N=compactEI(*M*)

N=

0 0 0 0
1 2 1 0
2 3 4 2
3 4 6 4
4 5 7 5

Where matrix *N* represents the *EI* element $m_1m_2m_3m_4+ m_2m_3m_4m_5+ m_1m_4m_6m_7+m_2m_4m_5$.

Program codes

```
function N=compactEI(M)
if any(M(1,:))
    error('the first row of input matrix must be zeros.')
end

Msize=size(M);
M=sort(M);
Max_Num_Column=0;

for i=1:Msize(1,2)
    Comp=unique(M(:,i));
    Csize=nnz(Comp);
    Max_Num_Column=max(Max_Num_Column,Csize(1));
    M(:,i)=[zeros((Msize(1,1)-Csize(1,1)),1);Comp(2:end)];
end

N=M((Msize(1)-Max_Num_Column):end,:);
return
```

2. compactEII

Compact an *EII* element.

Syntax

N=compactEII(*M*)

Description

N=compactEII(*M*) returns a matrix *N*, which represents an *EII* element in *EXM* equal to what is represented by *M*. The numbers of rows of *N* are the least of all matrices which represent the same *EII* element as *M*.

Examples

M=[0,1,2,3,3,4,0,3,3,4; 0,0,3,2,5,4,0,0,2,5; 0,1,1,6,7,4,0,1,4,7; 0,0,0,2,4,5,0,1,2,5]’

M=

0 0 0 0
1 0 1 0

2 3 1 0
3 2 6 2
3 5 7 4
4 4 4 5
0 0 0 0
3 0 1 1
3 2 4 2
4 5 7 5

N=compactEII(*M*)

N=

0 0 0 0
1 2 1 0
2 3 4 2
3 4 6 4
4 5 7 5
0 0 0 0
0 0 1 1
3 2 4 2
4 5 7 5

Where matrix *N* represents the *EII* element $\{x_3, x_4\}\{m_1, m_2, m_3, m_4\}+\{x_2, x_5\}\{m_2, m_3, m_4, m_5\}+\{x_1, x_4, x_7\}\{m_1, m_4, m_6, m_7\}+\{x_1, x_2, x_5\}\{m_2, m_4, m_5\}$.

Program codes

```
function N=compactEII(M)
Msize=size(M);
k=0;

for i=1:Msize(1)
    if (sum(M(i,:))~=0)
        break
    end
    k=k+1;
end

if k==0
    error('the first row of input matrix must be zeros.')
end

Mat=M(k:end,:);
EI_Num=find(sum(Mat,2)==0);

if size(EI_Num,1)==1
    error('the concept represented by input matrix must be a EII element.')
end
```

```
EI_Elem=Mat(1:EI_Num(2)-1,:);
Simple=Mat(EI_Num(2):end,:);
N=[compactEI(EI_Elem);compactEI(Simple)];
return
```

3. reduceEI

Reduce an *EI* element represented by an integer matrix

Syntax

N=reduceEI(*M*)

Description

N=reduceEI(*M*) returns the matrix *N*，which represents the most simple *EI* element in *EM* equal to what is represented by *M*. The result is sorted in ascending order.

Examples

M=[0,1,2,3,3,4; 0,0,3,2,5,4; 0,1,1,6, 7,4; 0,0,0,2,4,5]’

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2
3	5	7	4
4	4	4	5

Where matrix *M* represents the *EI* element $m_1m_2m_3m_3m_4+m_3m_2m_5m_4+m_1m_1m_6m_7m_4+m_2m_4m_5$.

N=reduceEI(*M*)

N=

0	0	0
0	1	1
2	2	4
4	3	6
5	4	7

Where matrix *N* represents the *EI* element $m_2m_4m_5+m_1m_2m_3m_4+m_1m_4m_6m_7$

According to Definitions1,the *EI* elements represented by *M* and *N* are equivalent.

Program codes

```
function N=reduceEI(M)

if any(M(1,:))
    error('the first row of input matrix must be zeros.')
end

M=compactEI(M);
M=tran_bool(M);
Reserve_EI=reduceEI_bool(M);
N=tran_EI_mat(Reserve_EI);
```

```
return
```

4. tran_bool

Transform the integer representations of the Elements in *EM* into Boolean value representation.

Syntax

N=tran_bool(*M*)

Description

N=tran_bool(*M*) returns the Boolean matrix *N*, which represents the same *EI* element as the integer matrix *M*.

Examples

M=[0,1,2,3,3,4; 0,0,3,2,5,4; 0,1,1,6,7,4; 0,0,0,2,4,5]’

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2
3	5	7	4
4	4	4	5

N=tran_bool(*M*)

N=

1	0	1	0
1	1	0	1
1	1	0	0
1	1	1	1
0	1	0	1
0	0	1	0
0	0	1	0

Where matrices *M*, *N* represent the same *EI* element $m_1m_2m_3m_4+m_2m_3m_4m_5+m_1m_4m_6m_7+m_2m_4m_5$.

Program codes

```
function N=tran_bool(M)
Num_Row=max(max(M));
Msize=size(M);
Receive=zeros(Num_Row,Msize(2));

for i=1:size(M,2)
    No_Zero_Row=find(M(:,i)~=0);
    No_Zero_Elem=M(No_Zero_Row,i);
    Receive(No_Zero_Elem ,i)=1;
end

N=Receive;
return
```

5. tran_EI_mat

Transform the Boolean value representations of the elements in *EM* into integer representations

Syntax

N=tran_EI_mat(*M*)

Description

N=tran_EI_mat(*M*) returns an integer matrix ,which represents the same *EI* element as Boolean matrix *M*.

Examples

M=[1 1 1 1 0 0 0; 0 1 1 1 1 0 0; 1 0 0 1 0 1 1; 0 1 0 1 1 0 0]'

M=

1	0	1	0
1	1	0	1
1	1	0	0
1	1	1	1
0	1	0	1
0	0	1	0
0	0	1	0

N=tran_EI_mat(*M*)

N=

0	0	0	0
1	2	1	0
2	3	4	2
3	4	6	4
4	5	7	5

Where matrices *M*,*N* represent the same *EI* element $m_1m_2m_3m_4+m_2m_3m_4m_5+m_1m_4m_6m_7+m_2m_4m_5$.

Program codes

```
function N=tran_EI_mat(M)
Set_M=1:size(M,1);
Repmat_Set_M=repmat(Set_M',1,size(M,2));
N=[zeros(1,size(M,2));Repmat_Set_M.*M];
N=compactEI(N);
return
```

6. reduceEI_bool

Reduce a element in *EM* represented by a Boolean matrix.

Syntax

N=reduceEI_bool(*M*)

Description

N=reduceEI_bool(*M*) returns the Boolean matrix *N*， which is the most simple form of the element in *EM* represented by *M*.

Examples

M= [1 1 1 1 0 0 0; 0 1 1 1 1 0 0; 1 0 0 1 0 1 1; 0 1 0 1 1 0 0]'

M=

1 0 1 0
1 1 0 1
1 1 0 0
1 1 1 1
0 1 0 1
0 0 1 0
0 0 1 0

N=reduceEI_bool(*N*)

N=

0 1 1
1 1 0
0 1 0
1 1 1
1 0 0
0 0 1
0 0 1

Where matrix *N* represents an *EI* element $m_2m_4m_5+m_1m_2m_3m_4+m_1m_4m_6m_7$.

Program codes

```
function N=reduceEI_bool(M)
Bool_Msize=size(M);
[Sum_Sort,Sum_Index]=sort(sum(M,1));

if Sum_Sort(1)==0
    N=0;
else
    M=M(:,Sum_Index);
    Reserve_EI=[];
    if Bool_Msize(2)==1
        N=M;
    else
        for i=1:Bool_Msize(1,2)
            if M(:,i)==0
                continue
            else
            Reserve_EI=[Reserve_EI,M(:,i)];
            Nnz_Element=find(M(:,i)~=0);
            Nnz_Sum=sum(M(Nnz_Element,:),1);
            Judge_EI=Nnz_Sum==Sum_Sort(i);
            Del_EI=find(Judge_EI==1);
            M(:,Del_EI)=0;
            end
        end
        N=Reserve_EI;
```

```
    end
end
return
```

7. reduceEII

Reduce an *EII* element

Syntax

N=reduceEII(*M*)

Description

N=reduceEII(*M*) returns the matrix *N*, which is the most simple form of the *EII* element represented by *M*.

Examples

M=[0,1,2,3,3,4,0,3,3,4; 0,0,3,2,5,4,0,0,2,5; 0,1,1,6,7,4,0,1,4,7; 0,0,0,2,4,5,0,1,2,5]’

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2
3	5	7	4
4	4	4	5
0	0	0	0
3	0	1	1
3	2	4	2
4	5	7	5

N=reduceEII(*M*)

N=

0	0	0
0	1	1
2	2	4
4	3	6
5	4	7
0	0	0
1	0	1
2	3	4
5	4	7

Where matrices *M*,*N* represent the same *EII* element $\{x_1, x_2, x_5\}\{m_2, m_4, m_5\}+\{x_3, x_4\}\{m_1, m_2, m_3, m_4\}+\{x_1, x_4, x_7\}\{m_1, m_4, m_6, m_7\}$.

According to Definitions 5, the *EII* elements represented by *M* and *N* are equivalent.

Program codes

```
function N=reduceEII(M)
if any(M(1,:))
    error('the first row of input matrix must be zeros.')
end
```

```
M=compactEII(M);

if size(M,2)==1
    N=M;
else
    Num_EI_Elem=find(sum(M,2)==0);
    Bool_EI_Elem=tran_bool(M(1:Num_EI_Elem(2)-1,:));
    Bool_EI_Elem_Size=size(Bool_EI_Elem);
    [Sum_Sort,Sum_Index]=sort(sum(Bool_EI_Elem,1));
    if Sum_Sort(1)==0
        N=M(:,Sum_Index(1));
    else
        Bool_EI_Elem=Bool_EI_Elem(:,Sum_Index);
        Simple=M(Num_EI_Elem(2):end,Sum_Index);
        Reserve_EI=[];
        Reduced_Simple=[];
        for i=1:Bool_EI_Elem_Size(1,2)
            if (Bool_EI_Elem(:,i)==0)
                continue
            else
            Reserve_EI=[Reserve_EI,Bool_EI_Elem(:,i)];
            Reduced_Simple=[Reduced_Simple,Simple(:,i)];
            Nnz_Element=find(Bool_EI_Elem(:,i)~=0);
            Nnz_Sum=sum(Bool_EI_Elem(Nnz_Element,:),1);
            Judge_EI=Nnz_Sum==Sum_Sort(i);
            Del_EI=find(Judge_EI==1);
            Bool_EI_Elem(:,Del_EI)=0;
            end
        end
        N=[tran_EI_mat(Reserve_EI);Reduced_Simple];
    end
end
return
```

8. reduceEIco

Reduce an $E^{\#}I$ element represented by an integer matrix

Syntax

N=reduceEIco(*M*)

Description

N=reduceEIco(*M*) returns the matrix *N*， which represents the most simple $E^{\#}I$ element in $E^{\#}X$ represented by *M*. The result is sorted in ascending order.

Examples

M=[0,1,2,3,3,4; 0,0,3,2,5,4; 0,1,1,6, 7,4; 0,0,0,2,4,5]’

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2
3	5	7	4
4	4	4	5

N=reduceEIco(*M*)

N=

0	0	0
1	2	1
2	3	4
3	4	6
4	5	7

Where matrices *M*, *N* represent the same $E^{\#}I$ element $x_1x_2x_3x_4+x_2x_3x_4x_5+x_1x_4x_6x_7$.

According to Definitions7 and Proposition2, the $E^{\#}I$ elements represented by *M* and *N* are equivalent.

Program codes

```
function N=reduceEIco(M)
if any(M(1,:))
    error('the first row of input matrix must be zeros.')
end
M=compactEI(M);
M=tran_bool(M);
Reserve_EI=reduceEIco_bool(M);
N=tran_EI_mat(Reserve_EI);
return
```

9. reduceEIco_bool

Reduce an $E^{\#}I$ element represented by a Boolean matrix

Syntax

N=reduceEIco_bool(*M*)

Description

N=reduceEIco_bool(*M*) returns the Boolean matrix *N*，which represents the most simple $E^{\#}I$ element in $E^{\#}X$ represented by Boolean matrix *M*.

Examples

M= [1 1 1 1 0 0 0; 0 1 1 1 1 0 0; 1 0 0 1 0 1 1; 0 1 0 1 1 0 0]’

M=

1	0	1	0
1	1	0	1
1	1	0	0
1	1	1	1
0	1	0	1

0 0 1 0

0 0 1 0

N=reduceEIco_bool(*N*)

N=

1 0 1

1 1 0

1 1 0

1 1 1

0 1 0

0 0 1

0 0 1

Where matrices *M*,*N* represent the same $E^{\#}I$ element $x_1x_2x_3x_4+x_2x_3x_4x_5+x_1x_4x_6x_7$.

Program codes

```
function N=reduceEIco_bool(M)
Bool_Msize=size(M);
[Sum_Sort,Sum_Index]=sort(sum(M,1),'descend');
M=M(:,Sum_Index) ;
Reserve_EI=[];
if Bool_Msize(2)==1
    N=M;
else
    for i=1:Bool_Msize(1,2)
        if M(:,i)==0
            continue
        else
        Reserve_EI=[Reserve_EI,M(:,i)];
        Zero_Element=find(M(:,i)==0);
        Zero_Sum=sum(M(Zero_Element,:),1);
        Judge_EI=Zero_Sum==0;
        Del_EI=find(Judge_EI==1);
        M(:,Del_EI)=0;
        end
    end
    N=Reserve_EI;
end
return
```

10. equal_EI

Judge the equivalence of two *EI* elements

Syntax

f=equal_EI(*M*,*N*)

Description

f=equal_EI(*M*,*N*) returns a Boolean value to judge the equivalence of two *EI* elements represented

by matrices *M*,*N*. If the two *EI* elements are equivalent, f=1, otherwise f=0.

Examples

M=[0 1 4 7; 0 0 1 4; 0 2 5 5]’

M=

0	0	0
1	0	2
4	1	5
7	4	5

N=[0 0 1 4; 0 2 5 5]’

N=

0	0
0	2
1	5
4	5

f=equal_EI(*M*,*N*)

f=1

Program codes

```
function f=equal_EI(M,N)
if less_EI(M,N)==1&less_EI(N,M)==1
    f=1;
else
    f=0;
end
return
```

11. less_EI

Judge if an *EI* element is less than the other

Syntax

f=less_EI(*M*,*N*)

Description

f=less_EI(*M*,*N*) returns a Boolean value to judge if an *EI* element represented by *M* is less than the one represented by *N*. If *M* is less, f=1, otherwise f=0.

Examples

M=[0 1 4 7; 0 2 5 5]’

M=

0	0
1	2
4	5
7	5

N=[0 0 1 4; 0 2 5 5]’

N=

0	0
0	2

1 5
4 5

f=less_EI(*M*,*N*)

f=1

Program codes

```
function f=less_EI(M,N)
Bool_M=tran_bool(M);
Bool_N=tran_bool(N);

Bool_M=reduceEI_bool(Bool_M);
Bool_N=reduceEI_bool(Bool_N);

Bool_Msize=size(Bool_M);
Bool_Nsize=size(Bool_N);

Size_Row=max(Bool_Msize(1),Bool_Nsize(1));
Bool_M=[Bool_M;zeros(Size_Row-Bool_Msize(1),Bool_Msize(2))];
Bool_N=[Bool_N;zeros(Size_Row-Bool_Nsize(1),Bool_Nsize(2))];

for i=1:Bool_Msize(2)
    Tran=repmat(Bool_M(:,i),1,Bool_Nsize(2));
    if (sum(Tran>=Bool_N,1)==Size_Row)==0
        f=0;
        break
    else
        f=1;
        continue
    end
end
return
```

12. equal_EIco

Judge the equivalence of two $E^{\#}I$ elements

Syntax

f=equal_EIco(*M*, *N*)

Description

f=equal_EIco(*M*, *N*) returns a Boolean value to judge the equivalence of two $E^{\#}I$ elements represented by matrices *M*,*N*. If the two $E^{\#}I$ elements are equivalent, f=1, otherwise f=0.

Examples

M=[0 1 4 7; 0 0 1 4; 0 2 5 5]’

M=

0 0 0
1 0 2

4 1 5
7 4 5

N=[0 1 4 7; 0 2 5 5]’

N=

0 0
1 2
4 5
7 5

f=equal_EIco(*M*,*N*)

f=1

Program codes

```
function f=equal_EIco(M,N)
if less_EIco(M,N)==1&less_EIco(N,M)==1
    f=1;
else
    f=0;
end
return
```

13. less_EIco

Judge if an $E^{\#}I$ element is less than the other

Syntax

f=less_EIco(*M*, *N*)

Description

f=less_EIco(*M*, *N*) returns a Boolean value to judge if an $E^{\#}I$ element represented by *M* is less than the other represented by *N*. If *M* is less, f=1, otherwise f=0.

Examples

M=[0 1 4 7; 0 2 5 5]’

M=

0 0
0 2
1 5
4 5

N=[0 0 1 4; 0 2 5 5]’

N=

0 0
1 2
4 5
7 5

f=less_EIco(*M*,*N*)

f=1

Program codes

```
function f=less_EIco(M,N)
Bool_M=tran_bool(M);
Bool_N=tran_bool(N);

Bool_M=reduceEIco_bool(Bool_M);
Bool_N=reduceEIco_bool(Bool_N);

Bool_Msize=size(Bool_M);
Bool_Nsize=size(Bool_N);

Size_Row=max(Bool_Msize(1),Bool_Nsize(1));
Bool_M=[Bool_M;zeros(Size_Row-Bool_Msize(1),Bool_Msize(2))];
Bool_N=[Bool_N;zeros(Size_Row-Bool_Nsize(1),Bool_Nsize(2))];

for i=1:Bool_Msize(2)
    Tran=repmat(Bool_M(:,i),1,Bool_Nsize(2));
    if (sum(Tran<=Bool_N,1)==Size_Row)==0
        f=0;
        break
    else
        f=1;
        continue
    end
end
return
```

14. EIsum

Find the disjunction of two *EI* elements

Syntax

f=EIsum(*M*,*N*)

Description

f=EIsum(*M*,*N*) returns a matrix which represents the disjunction of two *EI* elements *M*, *N*. And the result is reduced by reduceEI.

Examples

M=[0,1,2,3,3,4; 0,0,3,2,5,4; 0,1,1,6,7,4; 0,0,0,2,4,5]’

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2
3	5	7	4
4	4	4	5

N=[0,0,2,2,3,4; 0,1,4,5,6,4; 0,0,0,7,6,6]’

N=

0 0 0
0 1 0
2 4 0
2 5 7
3 6 6
4 4 6

f=EIsum(*M*,*N*)

f=

0 0 0 0
0 0 0 1
0 2 2 4
6 4 3 5
7 5 4 6

Where matrix *f* represents the *EI* element $m_6m_7+m_2m_4m_5+m_2m_3m_4+m_1m_4m_5m_6$, which is the disjunction of two *EI* elements *M*, *N* according to formula (1).

Program codes

```
function f=EIsum(M,N)
Msize=size(M);
Nsize=size(N);
f=reduceEI([[zeros(max(Msize(1),Nsize(1))-Msize(1),Msize(2));M],[zeros(max(Msize(1),Nsize(1))-Nsize(1),Nsize(2));N]]);
return
```

15. EIIsum

Find the disjunction of two *EII* elements

Syntax

f=EIIsum(*M*,*N*)

Description

f=EIIsum(*M*,*N*) returns a matrix which represents the disjunction of two *EII* elements *M*,*N*. And the result is reduced by reduceEII.

Examples

M=[0 1 2 3 4 0 4; 0 2 3 4 5 0 5]’

M=

0 0
1 0
2 2
3 3
0 0
0 4
4 5
3 3

N=[0 1 2 3 0 0 3 4; 0 0 2 3 0 3 4 5; 0 0 1 3 0 3 4 7]’

N=

0 0 0
1 0 0
2 2 1
3 3 3
0 0 0
0 3 3
3 4 4
4 5 7

f=EIIsum(*M*,*N*)

f=

0 0
2 1
3 3
0 0
3 3
4 4
5 7

Where matrix *f* represents the *EII* element $\{x_3, x_4, x_5\}\{m_2, m_3\}+\{x_3, x_4, x_7\}\{m_1, m_3\}$,which is disjunction of two *EII* elements *M*,*N* according to the formula (7).

Program codes

```
function f=EIIsum(M,N)
M=reduceEII(M);
N=reduceEII(N);

M_EI_Num=find(sum(M,2)==0);
N_EI_Num=find(sum(N,2)==0);

M_EI_Elem=M(1:M_EI_Num(2)-1,:);
N_EI_Elem=N(1:N_EI_Num(2)-1,:);
M_Simple=M(M_EI_Num(2):end,:);
N_Simple=N(N_EI_Num(2):end,:);

M_EI_Elem_Size=size(M_EI_Elem);
N_EI_Elem_Size=size(N_EI_Elem);
M_Simple_Size=size(M_Simple);
N_Simple_Size=size(N_Simple);

EI_Elem=[[zeros(max(M_EI_Elem_Size(1),N_EI_Elem_Size(1))-M_EI_Elem_Size(1),M_EI_Elem_Size(2));M_
EI_Elem],[zeros(max(M_EI_Elem_Size(1),N_EI_Elem_Size(1))-N_EI_Elem_Size(1),N_EI_Elem_Size(2));N_EI
_Elem]];
Simple=[[zeros(max(M_Simple_Size(1),N_Simple_Size(1))-M_Simple_Size(1),M_Simple_Size(2));M_Simple],[
zeros(max(M_Simple_Size(1),N_Simple_Size(1))-N_Simple_Size(1),N_Simple_Size(2));N_Simple]];
f=reduceEII([EI_Elem;Simple]);
```

```
return
```

16. EIcosum

Find the disjunction of two $E^{\#}I$ elements

Syntax

f=EIcosum(*M*, *N*)

Description

f=EIcosum(*M*, *N*) returns a matrix which represents the disjunction of two $E^{\#}I$ elements *M*,*N*. And the result is reduced by reduceEIco

Examples

M=[0,1,2,3,3,4; 0,0,3,2,5,4; 0,1,1,6, 7,4; 0,0,0,2,4,5]'

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2
3	5	7	4
4	4	4	5

N=[0,0,2,2,3,4; 0,1,4,5,6,4; 0,0,0,7,6,6]'

N=

0	0	0
0	1	0
2	4	0
2	5	7
3	6	6
4	4	6

f=EIcosum(*M*,*N*)

f=

0	0	0	0
1	2	1	1
2	3	4	4
3	4	6	5
4	5	7	6

Where matrix *f* represents the $E^{\#}I$ element $x_1x_2x_3x_4+x_2x_3x_4x_5+x_1x_4x_6x_7+x_1x_4x_5x_6$, which is the disjunction of two $E^{\#}I$ elements *M*, *N* according to Definition7 and formula (13) .

Program codes

```
function f=EIcosum(M,N)
f=reduceEIco([[zeros(max(size(M,1),size(N,1))-size(M,1),size(M,2));M],[zeros(max(size(M,1),size(N,1))-size(N,1),size(N,2));N]]);
return
```

17. EImult

Find the conjunction of two *EI* elements

Syntax

f=EImult(*M*,*N*)

Description

f=EImult(*M*,*N*) returns a matrix which represents the conjunction of two *EI* elements *M*, *N* .And the result is reduced by reduceEI

Examples

M=[0,1,2,3,3,4; 0,0,3,2,5,4; 0,1,1,6,7,4; 0,0,0,2,4,5]'

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2
3	5	7	4
4	4	4	5

N=[0,0,2,2,3,4; 0,1,4,5,6,4; 0,0,0,7,6,6]'

N=

0	0	0
0	1	0
2	4	0
2	5	7
3	6	6
4	4	6

f=EImult(M,N)

f=

0	0	0	0	0
0	0	0	2	1
2	1	1	4	2
3	2	4	5	4
4	3	6	6	5
5	4	7	7	6

Where matrix *f* represents the *EI* element $m_2m_3m_4m_5 + m_1m_2m_3m_4 + m_1m_4m_6m_7 + m_2m_4m_5m_6m_7 + m_1m_2m_4m_5m_6$, which is the conjunction of two *EI* elements *M*, *N* according to formula (2).

Program codes

```
function f=EImult(M,N)
Bool_M=tran_bool(M);
Bool_N=tran_bool(N);

Bool_M=reduceEI_bool(Bool_M);
Bool_N=reduceEI_bool(Bool_N);

Bool_Msize=size(Bool_M);
Bool_Nsize=size(Bool_N);
```

```
Bool_M=[Bool_M;zeros(max(Bool_Msize(1),Bool_Nsize(1))-Bool_Msize(1),Bool_Msize(2))];
Bool_N=[Bool_N;zeros(max(Bool_Nsize(1),Bool_Msize(1))-Bool_Nsize(1),Bool_Nsize(2))];

T=Bool_Msize(2)-Bool_Nsize(2);
Logicor=[];
if   T>0
     for i=1:Bool_Nsize(2)
          RepNi=repmat(Bool_N(:,i),1,Bool_Msize(2));
          Logicor=[Logicor,or(RepNi,Bool_M)];
          Logicor=reduceEI_bool(Logicor);
     end
else
     for i=1:Bool_Msize(2)
          RepMi=repmat(Bool_M(:,i),1,Bool_Nsize(2));
          Logicor=[Logicor,or(RepMi,Bool_N)];
          Logicor=reduceEI_bool(Logicor);
     end
end
f=tran_EI_mat(Logicor);
return
```

18. EIImult

Find the conjunction of two *EII* elements

Syntax

f=EIImult(*M*,*N*)

Description

f=EIImult(*M*,*N*) returns a matrix which is the conjunction of two *EII* elements *M*, *N* .And the result is reduced by reduceEII.

Examples

M=[0 1 2 3 4 0 4; 0 2 3 4 5 0 5]’

M=

```
     0     0
     1     2
     2     3
     3     4
     4     5
     0     0
     4     5
```

N=[0 1 2 3 0 0 3 4; 0 0 2 3 0 3 4 5; 0 0 1 3 0 3 4 7]’

N=

```
     0     0     0
     1     0     0
     2     2     1
     3     3     3
```

0 0 0
0 3 3
3 4 4
4 5 7

f=EIImult(*M*,*N*)

f=

0 0
1 2
2 3
3 4
4 5
0 0
4 5

Where matrix *f* represents the *EII* element $\{x_4\}\{m_1, m_2, m_3, m_4\}+\{x_5\}\{m_2, m_3, m_4, m_5\}$, which is conjunction of two *EII* elements *M*,*N* according to formula (8).

Program codes

```
function f=EIImult(M,N)
M=reduceEII(M);
N=reduceEII(N);

M_EI_Num=find(sum(M,2)==0);
N_EI_Num=find(sum(N,2)==0);

M_EI_Elem=M(1:M_EI_Num(2)-1,:);
M_Simple=M(M_EI_Num(2):end,:);
N_EI_Elem=N(1:N_EI_Num(2)-1,:);
N_Simple=N(N_EI_Num(2):end,:);

Bool_M_EI_Elem=tran_bool(M_EI_Elem);
Bool_N_EI_Elem=tran_bool(N_EI_Elem);
Bool_M_Simple=tran_bool(M_Simple);
Bool_N_Simple=tran_bool(N_Simple);

Logic_Elem=mult_bool(Bool_M_EI_Elem,Bool_N_EI_Elem,1);
Logic_Simple=mult_bool(Bool_M_Simple,Bool_N_Simple,2);
f=reduceEII([tran_EI_mat(Logic_Elem);tran_EI_mat(Logic_Simple)]);
return
```

19.mult_bool

Conjunction two Boolean matrices

Syntax

f=mult_bool(*M*, *N*, *Logic_Ind*)

Description

f=mult_bool(*M*, *N*, *Logic_Ind*) returns a matrix which is the conjunction of two Boolean matrices *M*, *N*. When the last argument: *Logic_Ind*=1, matrices *M*, *N* represent *EI* elements and the conjunction is obtained according to formula (2); *Logic_Ind*=2, matrices *M*, *N* represent $E^{\#}I$ elements and the conjunction is obtained according to formula (14).The result is not reduced.

Examples

M=[1 1 1 1 0 0 0; 0 1 1 1 1 0 0; 1 0 0 1 0 1 1]'

M=

1	0	1
1	1	0
1	1	0
1	1	1
0	1	0
0	0	1
0	0	1

N=[0 1 1 1 0 0; 1 0 0 1 1 1]'

N=

0	1
1	0
1	0
1	1
0	1
0	1

f=mult_bool(*M*, *N*, 1)

f=

1	0	1	1	1	1
1	1	1	1	1	0
1	1	1	1	1	0
1	1	1	1	1	1
0	1	0	1	1	1
0	0	1	1	1	1
0	0	1	0	0	1

Program codes

```
function f=mult_bool(M,N,Logic_Ind)
Msize=size(M);
Nsize=size(N);

M=[M;zeros(max(Msize(1),Nsize(1))-Msize(1),Msize(2))];
N=[N;zeros(max(Msize(1),Nsize(1))-Nsize(1),Nsize(2))];

T=Msize(2)-Nsize(2);
Logic=[];
if Logic_Ind==1
    if   T>=0
```

```
        for i=1:Nsize(2)
            RepNi=repmat(N(:,i),1,Msize(2));
            Logic=[Logic,or(RepNi,M)];
        end
    else
        for i=1:Msize(2)
            RepMi=repmat(M(:,i),1,Nsize(2));
            Logic=[Logic,or(RepMi,N)];
        end
    end
elseif Logic_Ind==2
    if  T>0
        for i=1:Nsize(2)
            RepNi=repmat(N(:,i),1,Msize(2));
            Logic=[Logic,and(RepNi,M)];
        end
    else
        for i=1:Msize(2)
            RepMi=repmat(M(:,i),1,Nsize(2));
            Logic=[Logic,and(RepMi,N)];
        end
    end
end
f=Logic;
return
```

20. EIcomult

Find the conjunction of two $E^{\#}I$ elements

Syntax

f=EIcomult(*M*, *N*)

Description

f=EIcomult(*M*, *N*) returns a matrix which is the conjunction of two $E^{\#}I$ elements *M*, *N* .And the result is reduced by reducecoEI.

Examples

M=[0,1,2,3,3,4; 0,0,3,2,5,4; 0,1,1,6,7,4; 0,0,0,2,4,5]’

M=

0	0	0	0
1	0	1	0
2	3	1	0
3	2	6	2
3	5	7	4
4	4	4	5

N=[0,0,2,2,3,4; 0,1,4,5,6,4; 0,0,0,7,6,6]’

N=

```
     0     0     0
     0     1     0
     2     4     0
     2     5     7
     3     6     6
     4     4     6
```

f=EIcomult(M,N)

f=

```
     0     0     0     0
     2     1     0     0
     3     4     4     6
     4     6     5     7
```

Where matrix *f* represents the $E^{\#}I$ element $x_2x_3x_4+x_1x_4x_6+x_4x_5+x_6x_7$, which is the conjunction of two $E^{\#}I$ elements *M*, *N* according to Definition 7 and formula (14).

Program codes

```
function f=EIcomult(M,N)
Bool_M=tran_bool(M);
Bool_N=tran_bool(N);

Bool_M=reduceEIco_bool(Bool_M);
Bool_N=reduceEIco_bool(Bool_N);

Bool_Msize=size(Bool_M);
Bool_Nsize=size(Bool_N);

Bool_M=[Bool_M;zeros(max(Bool_Msize(1),Bool_Nsize(1))-Bool_Msize(1),Bool_Msize(2))];
Bool_N=[Bool_N;zeros(max(Bool_Nsize(1),Bool_Msize(1))-Bool_Nsize(1),Bool_Nsize(2))];

T=Bool_Msize(2)-Bool_Nsize(2);
Logicor=[];
if   T>0
     for i=1:Bool_Nsize(2)
          RepNi=repmat(Bool_N(:,i),1,Bool_Msize(2));
          Logicor=[Logicor,and(RepNi,Bool_M)];
          Logicor=reduceEIco_bool(Logicor);
     end
else
     for i=1:Bool_Msize(2)
          RepMi=repmat(Bool_M(:,i),1,Bool_Nsize(2));
          Logicor=[Logicor,and(RepMi,Bool_N)];
          Logicor=reduceEIco_bool(Logicor);
     end
end
```

```
f=tran_EI_mat(Logicor);
return
```

21. negateEI

Find the negation of an *EI* element.

Syntax

N=negateEI(*M*)

Description

N=negateEI(*M*) find the logical negation of an *EI* element: *M*, according to formula (3), where m_{2*i} stands for the negation of the simple concept m_{2*i-1}.

Examples

M = [0,1,3,6; 0,0,5,9]’

M=

0	0
1	0
3	5
6	9

N=negateEI(*M*)

N=

0	0	0	0	0	0
2	4	5	2	4	5
6	6	6	10	10	10

Program codes

```
function N=negateEI(M)
Bool_M=tran_bool(M);
Bool_M=reduceEI_bool(Bool_M);
Msize=size(M);
Bool_Msize=size(Bool_M);

for i=1:Msize(2)
    for j=1:Msize(1)
      if M(j,i)==0
            continue
      elseif mod(M(j,i),2)==1
                M(j,i)=M(j,i)+1;
      else
              M(j,i)=M(j,i)-1;
      end
    end
end

T=M(:,1)';
Bool_N=tran_bool(T(:,find(T)));
```

```
for j=2:Bool_Msize(2)
    T=M(:,j)';
    Bool_N=mult_bool(reduceEI_bool(Bool_N),tran_bool(T(:,find(T))),1);
end

N=tran_EI_mat(reduceEI_bool(Bool_N));
return
```

22. gen_struture

Generate an *AFS* structure from the given original data and parameter matrix.

Syntax

Str_Mat=gen_structure(*Data_Mat*,*Parameter_Mat*)

Description

Str_Mat=gen_structure(*Data_Mat*,*Parameter_Mat*)returns an *AFS* structure (*M*, τ, *X*) by the semantic meanings with the parameters in *Parameter_Mat* and the original data in *Data_Mat*. The semantic meanings of the simple concepts in *M* are determined by the parameters in *Parameter_Mat*.

Examples

Data_Mat= [6.5,6.2,5.9,5.4]'

Parameter_Mat= [5.4,6.5,6.0]'

Str_Mat=gen_structure(*Data_Mat*,*Parameter_Mat*)

Str_Mat(:,:,1)=

1	1	1	1	1	1
0	1	1	0	0	1
0	1	1	0	0	1
0	1	1	0	1	0

Str_Mat(:,:,2)=

1	0	0	1	1	0
1	1	1	1	1	1
0	1	1	0	0	1
0	1	1	0	1	0

Str_Mat(:,:,3)=

1	0	0	1	1	0
1	0	0	1	1	0
1	1	1	1	1	1
0	1	1	0	1	0

Str_Mat(:,:,4)=

1	0	0	1	0	1
1	0	0	1	0	1
1	0	0	1	0	1
1	1	1	1	1	1

Program codes

```
function Str_Mat=gen_structure(Data_Mat,Parameter_Mat)
P_Mat_Size=size(Parameter_Mat);
D_Mat_Size=size(Data_Mat);

if P_Mat_Size(2)~=D_Mat_Size(2)
    error('the column numbers of the matrices do not match.')
end

Feature_Parameter_Index=~isnan(Parameter_Mat);
SC_Index=transform_simple_concepts_index(Parameter_Mat);

for i=1:D_Mat_Size(1)
    Repxi=repmat(Data_Mat(i,:),D_Mat_Size(1),1);
    for p=1:size(Feature_Parameter_Index,1)
        Is1_Index=find(Feature_Parameter_Index(p,:));
        Rep_PMatp=repmat(Parameter_Mat(p,Is1_Index), D_Mat_Size(1),1);
        Logicv=abs(Repxi(:,Is1_Index)-Rep_PMatp)<=abs(Data_Mat(:,Is1_Index)-Rep_PMatp);
        Neg_Logicv=abs(Repxi(:,Is1_Index)-Rep_PMatp)>=abs(Data_Mat(:,Is1_Index)-Rep_PMatp);
        Str_Mat(:,SC_Index(p,Is1_Index),i)=Logicv;
        Str_Mat(:,SC_Index(p,Is1_Index)+1,i)=Neg_Logicv;
    end
end
return
```

23. transform_simple_concepts_index

Transform a parameter matrix to simple concept indexes represented by a integer matrix

Syntax

SC_Index=transform_simple_concepts_index(*Parameter_Mat*)

Description

SC_Index=transform_simple_concepts_index(*Parameter_Mat*) returns a set of labels of the simple concepts in *M*. And the semantic meanings of the simple concepts in *M* are determined by the parameters in *Parameter_Mat*.

Examples

Parameter_Mat=[4.3, 5.8433, 7.9; NaN, 3.054, 4.4; NaN, NaN, 6.9; NaN,1.1987, 2.5]’

Parameter_Ma=

4.3	NaN	NaN	NaN
5.8433	3.054	NaN	1.1987
7.9	4.4	6.9	2.5

SC_Index=transform_simple_concepts_index(*Parameter_Mat*)

SC_Index=

1	0	0	0
3	7	0	13
5	9	11	15

Program codes

```
function SC_Index=transform_simple_concepts_index(Parameter_Mat)
C=~isnan(Parameter_Mat);
Csize=size(C);
SC_Index=zeros(Csize);
SC_Index(find(C==1))=2*[1:nnz(C)]-1;
Return
```

24. degree_xi

Find the membership degree on a fuzzy set

Syntax

f=degree_xi(*Lou_Mat*,*Str_Mat*,*Fuzzy_Set*,*Index_Xi*,*Logic_Ind*)

Description

f=degree_xi(*Lou_Mat*,*Str_Mat*,*Fuzzy_Set*,*Index_Xi*,*Logic_Ind*) returns the membership degree of the sample labeled by *Index_Xi* belonging to the fuzzy concept (in *EM*) represent by matrix *Fuzzy_Set* in the *AFS* structure represented by the 3-demension Boolean matrix *Str_Mat*. The weight functions of the simple concepts in M are represented by *Lou_Mat*, for detail, *Lou_Mat*(*i*, *j*)=$\rho_{mj}(x_i)$. Logic_Ind =1, the function returns the membership degree defined by formula (19); Logic_Ind =2, the function returns the membership degree defined by formula (20).

Examples

Str_Mat(:,:,1)=[1 1 1 1 1 1; 0 1 1 0 0 1; 0 1 1 0 0 1; 0 1 1 0 1 0]

Str_Mat(:,:,1)=

1	1	1	1	1	1
0	1	1	0	0	1
0	1	1	0	0	1
0	1	1	0	1	0

Str_Mat(:,:,2)=[1 0 0 1 1 0; 1 1 1 1 1 1; 0 1 1 0 0 1; 0 1 1 0 1 0]

Str_Mat(:,:,2)=

1	0	0	1	1	0
1	1	1	1	1	1
0	1	1	0	0	1
0	1	1	0	1	0

Str_Mat(:,:,3)=[1 0 0 1 1 0; 1 0 0 1 1 0; 1 1 1 1 1 1; 0 1 1 0 1 0]

Str_Mat(:,:,3)=

1	0	0	1	1	0
1	0	0	1	1	0
1	1	1	1	1	1
0	1	1	0	1	0

Str_Mat(:,:,4) =[1 0 0 1 0 1; 1 0 0 1 0 1; 1 0 0 1 0 1; 1 1 1 1 1 1]

Str_Mat(:,:,4) =

1	0	0	1	0	1
1	0	0	1	0	1
1	0	0	1	0	1
1	1	1	1	1	1

Lou_Mat=[0,1,1,0,0.1667,0.83333; 0.2727,0.7273,0.7273,0.2727,0.6667,0.3333; 0.5455,0.4545,0.4545,0.5455,0.8333,0.1667; 1,0,0,1,0,1]

Lou_Mat=

0	1	1	0	0.1666	0.8333
0.2727	0.7273	0.7273	0.2727	0.6667	0.3337
0.5455	0.4545	0.4545	0.5454	0.8333	0.1667
1	0	0	1	0	1

f=degree_xi(*Lou_Mat*,*Str_Mat*,[0 1; 0 3]',2,1)

f=0.5417

Program codes

```
function f=degree_xi(Lou_Mat,Str_Mat,Fuzzy_Set,Index_Xi,Logic_Ind)
Total=sum(Lou_Mat);
Result=0;
for k=1:size(Fuzzy_Set,2)
    Concept=Fuzzy_Set(:,k);
    Concept=Concept(find(Concept),:);
    if Logic_Ind==1
        A_Tao=under_A_xi(Str_Mat,Concept,Index_Xi);
        TT=sum(Lou_Mat(A_Tao,Concept))./Total(1,Concept);
        Result=max(Result,prod(TT));
    elseif Logic_Ind==2
        A_Tao=under_A_xi(Str_Mat,Concept,Index_Xi);
        TT=sum(Lou_Mat(A_Tao,Concept))./Total(1,Concept);
        Result=max(Result,min(TT));
    else
        TT=1;
        for i=1:size(Concept,1)
            T=find(Str_Mat(:,Concept(i),Index_Xi));
            TT=min(TT,sum(Lou_Mat(T,Concept(i)))/Total(1,Concept(i)));
        end
            Result=max(Result,TT);
    end
end
f=Result;
return
```

25. under_A_xi

Find the set of samples belonging to $A^{\tau}(x)$

Syntax

A_Tao=under_A_xi(*Str_Mat*,*Set_Simple_Concept*,*Index_Xi*)

Description

A_Tao=under_A_xi(*Str_Mat*,*Set_Simple_Concept*,*Index_Xi*) returns the labels of samples belonging to set $A^{\tau}(x_i)$ in *AFS* structure (M, τ, X), where the 3-dimension Boolean matrix *Str_Mat*

represents the *AFS* structure (M, τ, X). The column vector *Set_Simple_Concept* represents the labels of the simple concepts in set $A \subseteq M$. *Index_Xi* is the label of x_i.

Examples

Str_Mat(:,:,1)=[1 1 1 1 1 1; 0 1 1 0 0 1; 0 1 1 0 0 1; 0 1 1 0 1 0]

Str_Mat(:,:,1)=

1	1	1	1	1	1
0	1	1	0	0	1
0	1	1	0	0	1
0	1	1	0	1	0

Str_Mat(:,:,2)=[1 0 0 1 1 0; 1 1 1 1 1 1; 0 1 1 0 0 1; 0 1 1 0 1 0]

Str_Mat(:,:,2)=

1	0	0	1	1	0
1	1	1	1	1	1
0	1	1	0	0	1
0	1	1	0	1	0

Str_Mat(:,:,3)=[1 0 0 1 1 0; 1 0 0 1 1 0; 1 1 1 1 1 1; 0 1 1 0 1 0]

Str_Mat(:,:,3)=

1	0	0	1	1	0
1	0	0	1	1	0
1	1	1	1	1	1
0	1	1	0	1	0

Str_Mat(:,:,4) =[1 0 0 1 0 1; 1 0 0 1 0 1; 1 0 0 1 0 1; 1 1 1 1 1 1]

Str_Mat(:,:,4) =

1	0	0	1	0	1
1	0	0	1	0	1
1	0	0	1	0	1
1	1	1	1	1	1

Set_Simple_Concept=[0 1]'

Index_Xi=2

A_Tao=under_A_xi(*Str_Mat*,*Set_Simple_Concept*,*Index_Xi*)

A_Tao=

1

2

Program codes

```
function A_Tao=under_A_xi(Str_Mat,Set_Simple_Concept,Index_Xi)
Set_Simple_Concept=Set_Simple_Concept(find(Set_Simple_Concept),:);
A_Tao=find(sum(Str_Mat(:,Set_Simple_Concept,Index_Xi),2)==length(Set_Simple_Concept));
return
```