A

AGGREGATION IN FUZZY SYSTEMS AND SIMULATION OF NEURAL NETWORKS

by

ALEXANDER RYBALOV

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1995

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

8/16/85
Date

Chair of executive committee

Aug 15, 1985
Date

Executive Officer

Michael Anshel

Jacob Shapiro

Ronald R. Yager

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

AGGREGATION IN FUZZY SYSTEMS

AND SIMULATION OF NEURAL NETWORKS

by

Alexander Rybalov

Adviser: Professor Jerry Waxman

The central idea in artificial intelligence research has been that of finding adequate representations to model human thought. To this end, research has been focused on fuzzy logic. In this thesis a new kind of fuzzy logic operators - uni-norm operators - has been introduced, which can model more faithfully such logical connectives as conjunction and disjunction. Subsequently their properties have been investigated and the general representability was established.

Another type of aggregation in fuzzy systems arise in attempts to model the way in which humans process information. Such types of aggregation - full reinforcement aggregation and self identity operators - are introduced, and their application in fuzzy systems is investigated.

Information fusion deals with the problem of combining knowledge from different sources to obtain so called fused value. This work introduce the notion of a penalty function as a method for obtaining the fused value. A number of different penalty functions are investigated, and their applications are considered.

Fuzzy systems possess advantages in comparison with neural networks. In particular they consist of rules humans can understand, whereas neural networks are represented as impenetrable mass of weights. Putting neural networks in a form more amenable to human reasoning is a large step in understanding their work. The thesis shows how to represent neural networks through fuzzy systems.

Fuzzy systems are built by experts, whereas neural networks learn from examples. Enabling fuzzy systems also learn from examples greatly enhance their possibilities. In this work techniques used in building of neural networks - backpropagation method - is applied to fuzzy systems, so latter would be more consistent with both experts and examples.

ACKNOWLEDGEMENTS.

I like to thank my adviser, professor Jerry Waxman for his help in during my years in Graduate School. Despite his very tense schedule he found time to work with me.

I am very grateful to professor Ronald R. Yager for his participation in my scientific work. His insights were extremely clear and thoughtful.

I like to thank professor C.V. Negoita for the time he spent with me and for his suggestion concerning my thesis. He was the first who showed me the field which later became my specialization.

I also like to thank professors Michael Anshel and Jacob Shapiro for their help during my work on the dissertation.

Table of contents

Introduction.

A central idea of artificial intelligence research has been that of finding adequate representations to model human thought [Feienbaum, Feldman, 1993]. To this end, much research has focused on fuzzy logic. However, there are voices in the artificial intelligence community [Elkan, 1993] that assert that fuzzy logic is unable to adequately model human reasoning in certain crucial areas. In particular, Elkan states that fuzzy logic can't correctly model logical operators like conjunction and disjunction as used by humans. This thesis addresses this problem by introducing new kinds of operators which more faithfully model human thought. In chapter 1 we address this problem and extend fuzzy logic by introducing the concept of uni-norms to model adequately conjunction and disjunction. In chapter 2 properties of uni-norms are investigated, and the general representability of uni-norms is established.

In applications of fuzzy sets such as pattern recognition, information retrieval, diagnosis, expert systems, and multi-criteria decision making, aggregation often manifests itself in terms of the collection of scores drawn from the unit interval. These aggregation operators are required to possess a number of reasonable properties. Some of these properties are imposed by attempts to model the way in which humans process information. One characteristic of many types of human information processing is the tendency, on one hand, of a collection of high scores to reinforce each other to give a resulting score more affirmative than any of the individual scores alone, and on the other hand, the tendency of a collection of low scores to reinforce each other to give a resulting score more disfirmative than any of the individual scores alone. For example, in medical diagnosis, the appearance of a number of symptoms indicative of a disease will make us more confident in diagnosing a patient as having the disease than any one symptom alone, while the lack of appearance of the collection of these symptoms will make us more confident diagnosing a patient as not having a disease. In chapter 3, such type of aggregation is introduced as **full reinforcement aggregation**. In this chapter uni-norm aggregation operators as reinforcement operators are also studied.

Another type of aggregation arises in cases when if the value of a new element is equal to the result of the previous aggregation, then it doesn't influence the outcome of current

aggregation. For example, performance evaluation depends on aggregation of a number of different scores. If the latest score coincides with the aggregation of previous scores, the result should remain the same. The above requirement imposes a useful restriction on the weights associated with this aggregation, and permits one to consider a linear class of non-commutative aggregation operators. In chapter 5 this class is introduced and a number of special families of it are investigated.

Information fusion concerns itself with the problem of combining knowledge provided by different sources to obtain a so- called fused value [Abidi and Gonsales, 1992; Yager,1988]. In chapter 5, we introduce the notion of a **penalty function** as a method for obtaining the fused value. A number of different penalty functions are considered. Among these is the absolute difference between the observed value and the fused value. It is shown that in this case, the fused value is the median value of the observations. This approach is extended to the situation in which weights are associated with the observation and a formulation for a weighted median is obtained.

There are many applications of fuzzy systems. But in comparison with neural networks, they are not as widely used or well known. Neural networks have several advantages over fuzzy systems [Rich, Knight, 1991]: they can learn any function, they can separate any pattern classes; in theory they can model any system if they have enough neurons. But neural networks have one major drawback: their representation is an impenetrable mass of weights [Kosko, 1993], and therefore they are impossible to understand. If we can put them in a form which is more amenable to a human reasoning it would be a large step forward in the understanding of solutions offered by neural networks and their verifications and modifications. In contrast, fuzzy systems consist of rules humans can understand [Kosko, 1993]. In chapter 6, we show how to represent neural networks as fuzzy systems to enhance our understanding of what neural networks are actually doing. There is one more point to this: neural networks learn from examples, whereas fuzzy systems are built by experts. The question arises: can fuzzy systems also learn from examples? The answer is affirmative [Harris,1994]. However, the important problem is naintaining consistency between the rules of fuzzy systems which learn from examples consistent and experts' knowledge. We hold that to be consistent with experts' knowledge, fuzzy systems have to build fuzzy sets in such a

way that they differ but little from fuzzy sets used by experts. To deal with this problem we propose a new theoretical framework. To this end, we provide, in chapter 7, a new formalization of fuzzy systems. We apply classic techniques used by neural networks - the backpropagation method - to build the fuzzy systems which are consistent with both given examples and reasoning by experts. In addition, we later show that this approach can be used for the simulation of competitive neural networks via fuzzy systems. These issues are covered in chapter 7.

1. UNI-NORM LOGIC AND LOGIC OF FUZZY SYSTEMS.

1.1. Fuzzy logic and human reasoning.

Fuzzy logic is supposed to capture our reasoning about uncertainty. Therefore, conclusions about assertions with uncertain evidence in fuzzy logic should also be consistent with human reasoning. In works devoted to fuzzy logic this connection is often overlooked [Pedrycz, 1994], [McNeil, 1993]. This gives an opportunity for opponents of fuzzy logic to state that "fuzzy logic is not uniformly suitable for reasoning about uncertain evidence" [Elkan, 1993]. The above author presents the following argument:

"The universe of discourse is a collections of melons, and there are two predicates *red* and *watermelon*, where *red* and *green* refer to the colour of the flesh of a melon. From some not very well-known melon x suppose that $t(red(x)) = 0.5$ and $t(watermelon(x)) = 0.8$, meaning that the evidence that x is red inside has strength 0.5 and the evidence that x is watermelon has strength 0.8. According to the rules of fuzzy logic $t(red(x) \bigwedge watermelon(x)) = 0.5$. This is not reasonable, because watermelons are red inside. Redness and being a watermelon are mutually reinforcing facts, so intuitively, x is a red watermelon with certainty greater than 0.5."

"The deep issue here is that the degree of uncertainty of a conjunction is not in general determined uniquely by the degree of uncertainty of the assertions entering into the conjunction. There does not exist a function f such that the rule $t(A \bigwedge B) = f(t(A), t(B))$ is always valid, when t represents the degree of certainty of fragments of evidence. The certainty of $A \bigwedge B$ depends on the content of assertions A and B as well on their numerical certainty. This fact is recognized implicitly in probabilistic reasoning, since probability theory does not assign unique probability values to conjunctions. What probability theory says is that

$$1 - (1 - Pr(A) + 1 - Pr(B)) \leq Pr(A \bigwedge B) \leq \min\{Pr(A), Pr(B)\}.$$

The actual probability value depends on further aspects of the situation that have not been stated. For example, if the two assertions A and B are independent then the probability of

their conjunction is $Pr(A) \cdot Pr(B)$."

"Although probability theory is more flexible than fuzzy logic, the red watermelon example shows that it is not a universally adequate system of laws of thought for reasoning about all types of uncertainty either. If $t(red(x)) = 0.5$ and $t(watermelon(x)) = = 0.8$, then it is natural to want $t(red(x) \bigwedge watermelon(x)) > 0.5$, which probability theory can not permit."

The assumption underlying this example is that the function defining the degree of certainty of assertions of the conjunction is uniform across the interval [0,1]. If this assumption is valid then the above conclusion is correct. The argument is that the degree of certainty of the conjunction is more than the minimum of degrees of certainty of assertions if these degrees of certainty are no less than 0.5. This function behaves in two different ways when both arguments are no less than 0.5 and when both are no more than 0.5.

Thus, the function has some kind of fixed point (in this case 0.5), which defines its behavior. This fixed point intuitively corresponds to the notion that nothing definite is certain, that a melon could be red as well as non-red. Evidence that a melon is red has strength 0.5 does not affect our results. Thus, this point (0.5) corresponds to the **identity** of the operation. The operation on assertions whose strength is above the identity follows different rules than operation on numbers below the identity.

1.2. T-norm And T-conorm Operators And Reasoning About Uncertain Assertions.

This approach resembles some kind of generalization of *and* and *or* aggregation operators. There, the concepts of t-norm and t-conorm are particularly instrumental. A t-norm T is a mapping

$$T: [0,1] \times [0,1] \rightarrow [0,1]$$

with the following properties:

1) $T(a,b) = T(b,a)$	**Commutativity**
2) $T(a,b) \geq T(c,d)$ if $a \geq c$ and $b \geq d$	**Monotonicity**
3) $T(a,T(b,c)) = T(T(a,b),c)$	**Associativity**
4) $T(a,1) = a$	**Boundary**

It was shown in [Yager, 1993] that the t-norm operator has the property of anti-monotonicity in cardinality, $T(a_1,\ldots a_n) \geq T(a_1,\ldots a_{n+1})$. This property says that addition of elements to a t-norm aggregation can't result in an increase of the aggregate value.

A t-conorm S is a mapping

$$S: [0,1] \times [0,1] \rightarrow [0,1]$$

with the following properties:

1) $S(a,b) = S(b,a)$ **Commutativity**

2) $S(a,b) \geq S(c,d)$ if $a \geq c$ and $b \geq d$ **Monotonicity**

3) $S(a,S(b,c)) = S(S(a,b),c)$ **Associativity**

4) $S(a,0) = a$ **Boundary**

It was shown in [Yager, 1993] that a t-conorm operator has the property of monotonicity in cardinality, $S(a_1,\ldots a_n) \leq S(a_1,\ldots a_{n+1})$. This property says that the addition of elements to a t-conorm aggregation can't result in a decrease of the aggregate value.

The above example shows that when both elements are not less than 0.5, the conjunction in human reasoning behaves like a co-norm operator, whereas when both elements are not greater than 0.5, its behavior is like a t-norm operator. In this example, the function behaves like a co-norm operator; let us take the *max* operator as an example. The evidence that the melon is both red and a watermelon is $t(red(x) \bigwedge watermelon(x)) =$
$= \max(t(red(x)), t(watermelon(x))) = \max(0.5, 0.8) = 0.8$. This agrees with the intuition that evidence which is as likely as not to occur should be discarded.

1.3. Uni-norm operators And Combined Assertions.

Uni-norm operators are designed to capture human reasoning about assertions with uncertain evidence. The identity of these operators corresponds to assertions whose evidence is discarded. Other properties of uni-norm operators are the same as the properties of t-norm and t-conorm operators.

Definition 1.1: A uni-norm R is a mapping

$$R: [0,1] \times [0,1] \rightarrow [0,1]$$

with the following properties:

1) $R(a,b) = R(b,a)$ **Commutativity**

2) $R(a,b) \geq R(c,d)$ if $a \geq c$ and $b \geq d$ **Monotonicity**

3) $R(a,R(b,c)) = R(R(a,b),c)$ **Associativity**

4) There exists some element e in [0,1] called the **identity** such that for all a in [0,1] $R(a,e) = a$.

T-norm is a special case of uni-norm with identity one and t-conorm is a special case with identity zero.

It is easy to see that if both values are greater than or equal to identity than a t-norm operator is used, and if both values are less than or equal to identity then a t-conorm operator is used. It is possible to prove that among t-norm operators, the Min operator is the operator which gives the largest values for $a,b < e$.

Theorem 1.1: For any t-norm operator T and $a < b < e$

$$T(a,b) \leq a = \text{Min}(a,b).$$

Proof: Rewrite the second property (monotonicity) in

$$T(a,b) \leq T(a,e) = a.$$

The dual theorem is true for t-conorm.

Theorem 1.2: For any t-conorm operator S and $a > b > e$

$$S(a,b) \geq a = \text{Max}(a,b).$$

In the case when both values are greater than identity, assertions reinforce each other. But in human reasoning disjunction and conjunction work differently. In the case of conjunction the resulting assertion has the strength of the stronger argument. The *and* operator in human reasoning cannot make the strongest element stronger.

In the above example for $t(red(x)) = 0.6$ and $t(watermelon(x)) = 0.8$ with identity 0.5, the strength of the assertion $t(red(x) \bigwedge watermelon(x))$ is equal to 0.8. Thus, for assertions whose strength is greater than (or equal to) identity, the strength of conjunction is equal to the strength of the strongest assertion.

However, for the case of disjunction, the resulting assertion has strength more than the strongest argument. This means that adding an assertion results in increasing the strength, and, therefore, corresponds to a t-conorm.

If strengths of both assertions are less than identity, the strength of the conjunction of two assertions is strictly less than the strength of the weakest assertion. In the above example for $t(red(x)) = 0.4$ and $t(watermelon(x)) = 0.3$ with identity 0.5, the strength of the assertion $t(red(x) \bigwedge watermelon(x))$ is less than 0.3. In human reasoning in the conjunction (*and*) of two assertions, each of which is more likely not to be true than to be true, assertions weaken each other. Thus, adding an assertion corresponds to a t-norm operator. The strength of disjunction of two assertions, whose strength is less then identity, is equal to the strength of the strongest assertion. The resulting strength of the assertion $t(red(x) \bigvee watermelon(x))$ of the above example with $t(red(x)) = 0.4$ and $t(watermelon(x)) = 0.3$ and identity 0.5 is equal to $t(red(x)) = 0.4$. Therefore, the *max* operator corresponds to disjunction (*or*) when either of the assertions are more likely to be not true rather than true. This agrees with human intuition.

The last case to be considered is when one of the assertions has strength more than identity, and another has strength less than identity. In such a case, disjunction corresponds to the *max* operator, and conjunction to the *min* operator. In the former case, the reasoning behind this assignment is that the strength of the disjunction of two assertions cannot be less than the strength of either of them, whereas an assertion which is more likely to be untrue than true cannot increase the strength of the combining assertion (cf. Theorem 1.2). In the latter case (conjunction), the reasoning is that an assertion which is likely to be true (its strength is more than identity) cannot diminish the strength of the combined assertion. At the same time, the strongest assertion can't reinforce the weakest one since even a true assertion (whose strength is equal to 1) cannot enhance the assertion which likely is untrue (cf. theorem 1.1).

In sum, the above gives the following results:

Proposition 1.1: Uni-norm model for conjunction is (e is the identity)

a) $a,b \leq e$ $\quad t(a \bigwedge b) = T(a,b)$

where $T(a,b)$ is t-norm in the interval [0,e];

b) $a,b \geq e$ $\quad t(a \bigwedge b) = \max(a,b)$;

c) $a \leq e, b \geq e$ $\quad t(a \bigwedge b) = \min(a,b)$.

$b \leq e, a \geq e$

Proposition 1.2: Uni-norm model for disjunction is

a) $a,b \leq e$ $\quad t(a \vee b) = \max(a,b)$

b) $a,b \geq e$ $\quad t(a \vee b) = S(a,b)$;

where $S(a,b)$ is t-conorm in the interval [e,1];

c) $a \leq e, b \geq e$ $\quad t(a \vee b) = \max(a,b)$.

$b \leq e, a \geq e$

It is possible to represent both the conjunction and disjunction as different kinds of uni-norms. The resulting uni-norm should satisfy the above conditions, and, in addition, for each t-norm and t-conorm with standard identities (1 and 0) there is a corresponding t-norm and t-conorm with arbitrary identity e in the interval [0,1] (cf. section 2).

1.4 Modeling Uni-norm using T-norm and T-conorm Operators.

Existing t-norm and t-conorm operators can be used to model uni-norm operators. This makes itpossible to model both disjunction and conjunction as uni-norm operators.

Theorem 1.3: Suppose there is a t-norm T(a,b) with identity 1. The corresponding t-norm $T_e(a,b)$ for uni-norm with identity e for $a,b \leq e$ is equal $T_e(a,b) = e \cdot T(a/e,b/e)$.

Proof: It is necessary to check all properties.

Commutativity and monotonicity are obvious. For boundary condition

$$T_e(a,e) = e \cdot T(a/e,e/e) = e \cdot T(a/e,1) = e \cdot (a/e) = a$$

For associativity

$$T_e(T_e(a,b),c) = T_e(e \cdot T(a/b,b/e),c) = T(e \cdot T(a/e,b/e)/e,c/e) = T(T(a/e,b/e),c/e) =$$

$$= T(a/e,T(b/e,c/e)) = T(a/e,e \cdot T(b/e,c/e)/e) = T_e(a,e \cdot T(b/e,c/e)) = T_e(a,T_e(b,c)).$$

Theorem 1.4: Suppose there is t-conorm S(a,b) with identity 0. The corresponding t-conorm $S_e(a,b)$ for uni-norm with identity e for $a,b \geq e$ is equal

$S_e(a,b) = (1-e) \cdot S((a-e)/(1-e),(b-e)/(1-e))+e$.

Proof: As in the proof of theorem 3 it is necessary to check all properties. Commutativity and monotonicity are obvious. For the boundary condition

$$S_e(a,e) = (1-e) \cdot S((a-e)/(1-e),(e-e)/(1-e))+e =$$

$= (1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),0)+e = (1\text{-}e)\cdot((a\text{-}e)/(1\text{-}e))+e =$

$= a\text{-}e+e = a.$

For associativity

$S_{\bullet}(S_{\bullet}(a,b),c) = S_{\bullet}((1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),(b\text{-}e)/(1\text{-}e))+e,c)$

$= (1\text{-}e)\cdot S((1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),(b\text{-}e)/(1\text{-}e))+e\text{-}e)/(1\text{-}e),(c\text{-}e)/(1$

$\text{-}e))+e =(1\text{-}e)\cdot S((1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),(b\text{-}e)/(1\text{-}e))/(1\text{-}e),(c$

$\text{-}e)/(1\text{-}e))+e = (1\text{-}e)\cdot S(S((a\text{-}e)/(1\text{-}e),(b\text{-}e)/(1\text{-}e)),(c\text{-}e)/(1\text{-}e))+e$

$= (1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),S((b\text{-}e)/(1\text{-}e)),(c\text{-}e)/(1\text{-}e)))+e =$

$=(1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),(1\text{-}e)\cdot S((b\text{-}e)/(1\text{-}e))/(1\text{-}e),(c\text{-}e)/(1\text{-}e)))+e =$

$=(1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),(1\text{-}e)\cdot S((b\text{-}e)/(1\text{-}e))/(1\text{-}e),(c\text{-}e)/(1\text{-}e))\text{-}e+e)+e$

$= (1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),(1\text{-}e)\cdot S((b\text{-}e)/(1\text{-}e))/(1\text{-}e),(c\text{-}e)/(1\text{-}e))+e\text{-}e)+e$

$= (1\text{-}e)\cdot S((a\text{-}e)/(1\text{-}e),S_{\bullet}((b\text{-}e)/(1\text{-}e)),(c\text{-}e)/(1\text{-}e))\text{-}e)+e = S_{\bullet}(a,S_{\bullet}(b,c)).$

Proposition 1.1 and proposition 1.2 together with theorem 3 and theorem 1.4 provide means to model disjunction and conjunction which are closer to ones used in human reasoning.

Example: The product $a\cdot b$ can be taken as an example of a t-norm, and the Lucasevich operator $a+b\text{-}a\cdot b$ as an example of a t-conorm.

The uni-norm operator for disjunction is

a) $a,b \le e$ $\quad t(a \vee b) = \max(a,b)$;

b) $a,b \ge e$ $\quad t(a \vee b) = a+b\text{-}e\text{-}(a\text{-}e)\cdot(b\text{-}e)/(1\text{-}e)$;

c) $a \le e,\ b \ge e$ $\quad t(a \vee b) = \max(a,b)$.

$b \le e,\ a \ge e$

The uni-norm operator for conjunction is

a) $a,b \le e$ $\quad t(a \wedge b) = a\cdot b/e$;

b) $a,b \ge e$ $\quad t(a \wedge b) = \max(a,b)$;

c) $a \le e,\ b \ge e$ $\quad t(a \wedge b) = \min(a,b)$.

$b \le e,\ a \ge e$

2. Uni-Norm Aggregation operators

2.1 Properties of Uninorms

A kind of de Morgan duality exists for these uni-norm operators, viz:

Theorem 2.1: Assume R is a uni-norm with identity element e, then $\hat{R}$, defined such that

$$\hat{R}(a, b) = 1 - R(\bar{a}, \bar{b})$$

where $\bar{a} = 1 - a$, is a uni-norm with identity $\bar{e} = 1 - e$

Proof: Commutativity: This follows directly from the commutativity of R.

Monotonicity: Assume $a \geq c$ and $b \geq d$ then since $\bar{c} \geq \bar{a}$ and $\bar{d} \geq \bar{b}$ it follows that $R(\bar{c}, \bar{d}) \geq R(\bar{a}, \bar{b})$. Hence $1 - R(\bar{a}, \bar{b}) \geq 1 - R(c, d)$ and finally $\hat{R}(a, b) \geq \hat{R}(c, d)$

Associativity: $\hat{R}(a, \hat{R}(b, c)) = \hat{R}(a, 1 - R(\bar{b}, \bar{c})) = 1 - R(\bar{a}, R(\bar{b}, \bar{c})) = 1 - R(R(\bar{a}, \bar{b}), \bar{c})$

$$= 1 - R(1 - \hat{R}(a, b), \bar{c})) = \hat{R}(\hat{R}(a, b), c)$$

Identity: Since $\hat{R}(a, \bar{e}) = 1 - R(\bar{a}, e) = 1 - \bar{a} = a$, $\bar{e}$ is an identity.

The following lemma will be useful.

Lemma 2.1: Assume R is a uni-norm with identity e then

1. For any a and all $b > e$ we get $R(a, b) \geq a$
2. For any a and all $b < e$ we get $R(a, b) \leq a$

Proof: From the identity property we get $R(a, e) = a$. Consider $R(a, b)$; if $b > e$ then from monotonicity we get $R(a, b) \geq R(a, e) \geq a$. Similarly, if $b < e$ then again from monotonicity we get $R(a, b) \leq R(a, e) \leq a$.

The next theorem shows the role that e plays in effecting the aggregation.

Theorem 2.2: Assume R is a uni-norm with identity e, then

1. if $a_{n+1} < e$

$$R(a_1, \ldots, a_n) \geq R(a_1, \ldots, a_n, a_{n+1})$$

2. if $a_{n+1} > e$

$$R(a_1, \ldots, a_n) \leq R(a_1, \ldots, a_n, a_{n+1})$$

Proof: Assume $R(a_1, \ldots a_n) = a$; then from the associativity property of R we have

$$R(a_1, \ldots, a_n, a_{n+1}) = R(a, a_{n+1}).$$

and the result from the lemma.

Two special cases are worth noting. If e = 0, then every $a_{n+1} \geq e$, and hence the addition of an element tends to increase our aggregation. Similarly, if e = 1, then every $a_{n+1} \leq e$ and hence the addition of an element can never increase our aggregation.

The property of associativity allows us to extend the uni-norm from an operator with two arguments to one with n > 2 arguments. It is interesting to consider the uni-norm operator on less than two arguments. We first note that

$$R(a_1, \ldots a_n, e) = R(a_1, \ldots a_n)$$

so that the addition of the identity should not affect the value. If we consider R(a) we can always say

$$R(a) = R(a, e).$$

Furthermore, since R(a, e) = a, it follows that

$$R(a) = a.$$

Thus the uni-norm with one argument yields just that one argument as its value.

Consider now the null aggregation, i.e. R(). Since from consistency

$$R() = R(e) = R(e, e) = e$$

we have that R() = e. This is consistent with the use of t-norms and t-conorms, where T() is assumed to equal 1 and S() is assumed to equal zero.

An interesting property associated with t-norms and t-conorms is the boundary property, i.e. for any a, T(a, 0) = 0 and S(a, 1) =1.. The next theorem generalizes the boundary property to uni-norm operators.

Theorem 2.3: Assume R is a uni-norm with identity e. Then

1. $R(a, 0) = 0$ for all $a \leq e$
2. $R(a, 1) = 1$ for all $a \geq e$

Proof: (1) For $a \leq e$, $R(a, 0) \leq R(e, 0) = 0$

(2) For $a \geq e$, $R(a, 1) \geq R(e, 1) = 1$

Corollary: If e = 0, R(a, 1) = 1 for all a, this is the case of a t-conorm.

Corollary: If e = 1, R(a, 0) = 0 for all a, this is the case of t-norms.

2.3. A Family of Uni-norms

While the literature clearly supports the existence of an infinite number of uni-norms with identity zero or one, i.e. t-conorms and t-norms, the "instantiation" of uni-norms for other values of the identity is not that obvious. In this section we establish a class of uni-norm operators that have identities drawn from the unit interval.

Theorem 2.4: The operator R_*: $[0, 1] \times [0, 1] \rightarrow [0, 1]$ defined such that

(1) For $a, b \leq e$ $(Max(a, b) \leq e)$

$$R_*(a, b) = Min(a, b)$$

(2) For $a, b \geq e$ $(Min(a, b) \geq e)$

$$R_*(a, b) = Max(a, b)$$

(3) For $Max(a, b) > e$ and $Min(a, b) < e$

$$R_*(a, b) = Min(a, b)$$

is a uni-norm operator with identity e.

Proof:(1) Commutativity: This follows from the use of Max and Min operators which are commutative

(2) Monotonicity: Consider R(a, b) and R(c, d) where $a \geq c$ and $b \geq d$. R(a, b) = Max(a, b) or Min(a, b) and R(c, d) = Max(c, d) or Min(c, d)

i) Assume R(a, b) = Max(a, b). Since $Max(a, b) \geq Max(c, d)$ and $Max(a, b) \geq Min(c, d)$ we have $R(a, b) \geq R(c, d)$.

ii) Assume R(a, b) = Min(a, b). This implies that $Min(a, b) < e$. This in turn implies that $Min(c, d) < e$ and hence R(c, d) = Min(c, d) therefore $R(a, b) \geq R(c, d)$.

(3) Identity: Consider $R_*(a, e)$:

i) if $a \leq e$ then $R_*(a, e) = Min(a, e) = a$.

ii) if $a > e$ then $R_*(a, e) = Max(a, e) = a$.

(4) Associativity: Here we must establish the relationship

$$R_*(a, R_*(b, c)) = R_*(R_*(a, b), c)$$

Without loss of generality assume $a \geq b \geq c$.

i) Assume $e \leq c$. In this case

$$R_*(b, c) = \text{Max}(b, c) \geq e \text{ and } R_*(a, b) = \text{Max}(a, b) \geq e$$

hence

$$R_*(a, R_*(b, c)) = R_*(a, b) = \text{Max}(a, b) = a$$

and

$$R_*(R_*(a, b), c) = R_*(a, c) = \text{Max}(a, c) = a$$

ii) Assume $e \geq a$. In this case

$$R_*(b, c) = \text{Min}(b, c) = c$$

$$R_*(a, b) = \text{Min}(a, b) = b$$

$$R_*(R_*(a, b), c) = R_*(b, c) = \text{Min}(b, c) = c$$

$$R_*(a, R_*(b, c)) = R_*(a, c) = \text{Min}(a, c) = c$$

iii) Assume $a \geq e \geq b \geq c$. In this case

$$R_*(a, R_*(b, c)) = R_*(a, \text{Min}(b, c)) = R_*(a, c) = \text{Min}(a, c) = c$$

$$R_*(R_*(a, b) , c) = R_*(\text{Min}(a, b), c) = R_*(b, c) = \text{Min}(b, c) = c$$

iv) Assume $a \geq b \geq e \geq c$. In this case

$$R_*(a, R_*(b, c)) = R_*(a, \text{Min}(b, c)) = R_*(a, c) = c$$

$$R_*(R_*(a, b), c) = R_*(\text{Max}(a, b), c) = R_*(a, c) = c. \text{ V}$$

We should note that any attempt to try to generalize this family of uni-norms by replacing Min and Max with other t-norms and t–conorms runs into a problem in that we require $T(a, e) = a$ and $S(a, e) = a$ in order to satisfy the identity requirement.

A more concise characterization of the uni-norm R_* is

(1) $R_*(a, b) = \text{Min}(a, b)$ if $\text{Min}(a, b) < e$

(2) $R_*(a, b) = \text{Max}(a, b)$ if $\text{Min}(a, b) \geq e$

This implies that if one of the arguments is less than e then we use the minimum.

There exists another family of uni-norms closely related to to R*.

Theorem 2.5: The operator R* defined such that

(1) For a, b ≤ e

$$R^*(a, b) = \text{Min}(a, b)$$

(2) For a, b ≥ e

$$R^*(a, b) = \text{Max}(a, b)$$

(3) For Max(a, b) > e and Min(a, b) < e

$$R^*(a, b) = \text{Max}(a, b)$$

is a uni-norm operator with identity e.

We see that the difference between these two operators is in the third case.

An alternative characterization of R* is

(1) $R^*(a, b) = \text{Min}(a, b)$ if $\text{Max}(a, b) \leq e$

(2) $R^*(a, b) = \text{Max}(a, b)$ if $\text{Max}(a, b) > e$

The following two theorems help characterize these new classes of operators.

Theorem 2.7: If $\text{Min}(a_1, \ldots a_n) < e$ then

$$R_*(a_1, \ldots, a_n) = \text{Min}(a_1, \ldots, a_n)$$

else

$$R_*(a_1, \ldots, a_n) = \text{Max}(a_1, \ldots, a_n)$$

This theorem indicates that if at least one element lies below e then use the minimum operator.

Theorem 2.8: If $\text{Max}(a_1, \ldots a_n) > e$ then

$$R^*(a_1, \ldots, a_n) = \text{Max}(a_1, \ldots, a_n)$$

else

$$R^*(a_1, \ldots, a_n) = \text{Min}(a_1, \ldots, a_n)$$

Thus for R* if at least one element lies above e then use the maximum operator.

Figure 2.1 and Figure 2.2 help illustrate the functioning of these aggregation operators.

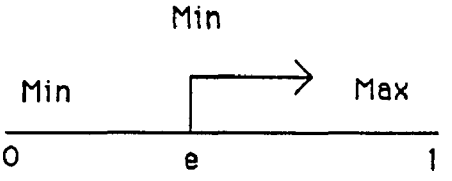

Figure #2.1: R_*

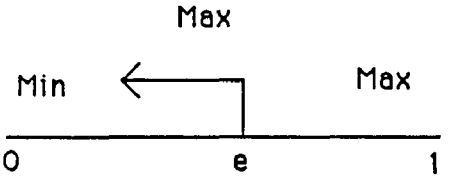

Figure #2.2: R*

Figures 1 and 2 indicate that if all the elements are to the left of e then use Min, and if all the elements are to the right then use Max. In Figure 2.1, R_*, we mean to indicate that if the elements straddle the identity element then use Min, in Figure 2.2, R*, we mean to indicate that if the elements straddle the identity element then use Max.

In order to get a better understanding of the semantics underlying these operators we shall describe their functioning in the framework of multi-criteria aggregation.

Consider first the case of R_*. We first note that for e = 1 this constitutes a pure t-norm while for e = 0 it constitutes a pure t-conorm. Let us start with e = 1. Since everything is to the left of e we always use the Min operator. Consider the situation as we move the value of e from one to something less than but still close to one (see Figure 2.3).

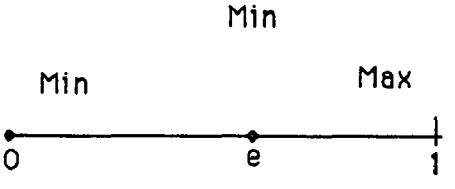

Figure #2.3: R_* for variable e

Here we see that if <u>any</u> of the components lie to the left of e, then we use the Min; otherwise we use the Max. We see that this is still fundamentally an *and* aggregation in that since the interval

[0, e] is large we shall most likely get some argument of the aggregation in that range. In this situation we see that e is some level such that if all the criteria are satisfied to at least e-extent, then we are satisfied and can measure our satisfaction by the best of the elements. On the other hand if any element goes below e then we take the worst. A semantics for this situation is

if all criteria are at least e then I am satisfied with any of them else I want all of them satisfied.

We see that as e gets smaller and smaller, it becomes less and less likely that we shall have an element in the left region and we begin to approach a pure *or*. If we consider the arguments of the aggregation as being randomly drawn from the unit interval we can see that for R_* the probability it will be a pure *and* (Min operator) is

$$P(R_* = and) = e \sum_{i=0}^{n-1} e^i = 1 - \bar{e}^n$$

where n is the number of arguments.

We can use the above as a measure of *andness* of the aggregation R_*. Here the measure is both a function of e and the number of elements. We see that as $n \to \infty$ or $e \to 1$ we become more certain it will act as an *and*.

We can measure the *orness* of R_* using $\bar{e}^n$. From Figure 3 we see that as e approaches zero the operator becomes very *orlike*. As a matter of fact, if e = 0, then this operator acts exactly like a Max (or). Again we note that the functioning of e is like a threshold; any satisfaction below this threshold turns R_* into a minimizer while all satisfaction above e turns it into a maximizer.

Let us now consider R*. Again we note that for e = 1 this is a pure t-norm and for e = 0 it is a pure t-conorm. Consider the situation for e different from 1 or 0 (see Figure 2.4).

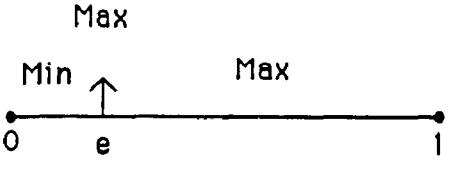

Figure #2.4.

Here we see that if any of the arguments are to the right of e, then we use Max, otherwise we use Min. For e close to zero R* is essentially acting like an *or*, as e moves closer to one R* begin to approach an *and* operator.

Again, we can introduce a measure of *andness/orness* associated with this operator. If we consider an argument randomly drawn from the unit interval

$$P(R^* = or) = 1 - e^n$$

Thus, e^n is the probability it will be a pure *and*. Thus

$$orness(R^*) = 1 - e^n$$

$$andness(R^*) = e^n$$

This can be compared with R_* where

$$orness(R_*) = \bar{e}^n$$

$$andness(R_*) = 1 - \bar{e}^n$$

A semantics that can be associated with the R* operator is

if any of the criteria are above e then I am satisfied with any of them else I want them all satisfied.

2.3. Weighted Uni-Norm Aggregation

In the preceding we have introduced the concept of uni-norm aggregators, each of which has an associated identity element e, and we have produced two families of these operators. In this section we shall introduce the notion of weighted uni-norm aggregations which can be used in situations in which each of the arguments (the a_i's) have an associated weight indicating the importance or relevance of the argument to the aggregation.

The weighted uni-norm aggregation operator can be implemented as follows. Assume R is uni-norm aggregator with identity e. Assume we have a collection of n pairs (a_i, w_i), with an argument value a_i and a weight w_i, both of which are contained in the unit interval. To find the weighted aggregation of $R((a_i, w_i))$ we proceed as follows:

1. Transform the pair (a_i, w_i) into a single value b_i, called its transformed value, where

$$b_i = g(w_i, a_i)$$

2. Calculate the uni-norm aggregation of the transformed values

$$R(b_1, \ldots b_n).$$

The function g which maps a weight and an argument value into a transformed value is

required to satisfy the following four conditions [Yager, 1993]:

I. Monotonicity in value: For $a > a'$ we require that

$$g(w, a) \geq g(w, a').$$

This condition essentially says that as the value increases the transformed value should follow.

II. Zero importance elements should have no effect. We require therefore, that

$$g(0, a) = e,$$

where e is the identity of the aggregation R. Here we see that if the object has zero importance then it gets transformed into the identity of the aggregation R and has no effect on the aggregate value.

III. Normality of importance of one, i.e.,

$$g(1, a) = a.$$

Thus the effect of g on a_i when $w_i = 1$ is to simply return a_i.

IV. Consistency of effect of w_i. Here we mean to say that as w goes from 0 to 1, g(w, a) monotonically moves from e to a. Thus

$$\text{for } x \geq y \qquad g(x, a) \geq g(y, a) \qquad \text{if } a \geq e$$

$$\text{for } x \geq y \qquad g(x, a) \leq g(y, a) \qquad \text{if } a \leq e$$

We note that if g is differentiable this last condition says

$$\frac{\partial g}{\partial x}(x, a) \geq 0 \qquad \text{if } a > e$$

$$\frac{\partial g}{\partial x}(x, a) \leq 0 \qquad \text{if } a < e$$

One possible formulation for the transformation operation g introduced above is

$$g(w, a) = wa + \overline{w}e.$$

It is easy to see that this satisfies the above four conditions. The monotonicity with respect to a is obvious. Zero importance, $w = 0$, gives us $g(w, a) = e$. If $w = 1$ then $\overline{w} = 0$ and hence $g(w, a) = a$. Finally for the last condition

$$\frac{\partial(w, a)}{\partial w} = a - e$$

We see that if $a \geq e$ then the derivative is positive, else it is negative.

We should note that for $e = 0$, we get

$$g(w, a) = wa$$

and for e = 1, we get

$$g(w, a) = wa + \overline{w}.$$

Furthermore the later case can be expressed as

$$g(w, a) = \overline{w} + a - (\overline{w}a).$$

The following example illustrates the effect of this formulation of g.

Example: Assume e = .5. Consider $R_*(a_1, a_2, a_3, a_4)$ where

$$a_1 = .7, a_2 = .8, a_3 = .3, a_4 = .9$$

Since $\exists$ a_i such that $a_i < .5$, then $R_*(a_i) = \text{Min}(a_i) = .3$.

Assume we now have weights associated with these arguments

$$w_1 = 0, w_2 = 1, w_3 = .2, w_4 = .7.$$

Using the transformation function g previously introduced we get

$$b_i = w_i a_i + \frac{1}{2}\overline{w}_i$$

and hence

$$b_1 = \frac{1}{2}(1) = \frac{1}{2}$$
$$b_2 = (1)(.8) = .8$$
$$b_3 = (.3)(.2) + \frac{1}{2}(.8) = .46$$
$$b_4 = (.9)(.7) + \frac{1}{2}(.3) = .63 + .15 = .78$$

In this new situation we desire now to form $R_*(b_i)$. Since .46 < .5 we get

$$R_*(b_i) = .46.$$

We see that since a_3 has small importance g moves its transformed value closer to the identity.

Let us look more generally at the effect of the transformation under consideration, viz.

$$b = wa + \overline{w}e.$$

First assuming a < e, we see that

$$b = wa + \overline{w}e \leq e$$

for all w. Similarly if a > e

$$b = wa + \overline{w}e \geq e$$

for all w. Thus the transformation never moves an element from one side of e to the other.

Consider the absolute difference of e and b,

$$|e - b| = |e - (wa + (1 - w)e)| = |e - wa - e + we| = w|e - a|$$

It is apparent that the distance between the transformed value b and e shrinks as w gets smaller; the lessening of importance of an argument brings its transformed value closer to the identity.

One can view the reasoning process used in fuzzy logic controllers as a kind of weighted uni-norm aggregation [Yager, 1993]. In this environment we have a collection of rules of the form

$$\text{If U is } A_i \text{ then V is } B_i$$

as well as a value for the antecedent variable, say, U is x*. This input value provides a firing level $\tau_i = A_i(x^*)$ for each of the individual rules. The problem, then is to aggregate the effects of each of the individual rules to give us an overall system output. One can consider that τ_i provides a measure of relevancy (importance) for each rule. Thus if we denote F as the fuzzy subset that is the overall output of the system it is obtained as a weighted uni-norm aggregation of the B_i (where τ_i is the weight of B_i in this aggregation). Formally, with R being a uni-norm, we can express this as

$$F(y) = R((\tau_i, B_i(y))).$$

If we select R with e = 0, then we get

$$F(y) = \text{Max}_i[\tau_i * B_i(y)]$$

which is an example of the kind aggregation used by Mamdani in his pioneering work on fuzzy logic control.[Mamdani, Assilian, 1975]

If we select an R with e = 1 then we get

$$F(y) = \text{Min}_i[\tau_i * B_i(y) + \bar{\tau}_i]$$

$$= \text{Min}_i[\bar{\tau}_i + B_i(y) - B_i\,\bar{\tau}_i]$$

This form for F(y) is a kind of logical interpretation; it is a form of implication and was introduced by Zadeh. [Zadeh, 1983]

2.4. Generalizations of R-Star

In an earlier section we introduced the concept of a uni-norm, which we defined as an operator which is commutative, associative, monotone, and with an identity element $e \in [0, 1]$. Furthermore, we suggested two classes of uni-norm aggregators, R^* and R_* which we shall now refer to collectively as R-star aggregators. These R-star operators are parameterized by their identity value e.

In [Yager, 1993] Yager introduced a class of operators which he called MICA operators. These operators are commutative, monotone and admit an identity, but they don't require the property of associativity. For many applications associativity is not a necessary requirement; the simple average, for instance is not associative. In [Yager, 1993] Yager showed that MICA operators constitute the basic operators needed for aggregation in fuzzy system modeling.

It is interesting to note that the property of associativity serves two purposes. First it allows us to simplify the aggregation in the following way,

$$\text{if } Agg(x_1, \ldots, x_n) = a \text{ then } Agg(x_1, \ldots x_n, x_{n+1}) = Agg(a, x_{n+1})$$

Second, it helps us simplify the definition of an operation by permitting the use of the binary definition of the operation to define the operation for multi arguments.

In this section we shall suggest some generalizations of the R-star operator which are not necessarily uni-norms but are MICA operators. The basic spirit of these R-star aggregators is that the identity value e acts as a boundary; if all elements are above the boundary we take the Max, and if all elements are below the boundary we take the Min and if elements straddle the boundary we take Max for R^* and the Min R_*. While these generalizations we are about to suggest may not capture all the properties of the uni-norm they capture the essential features of the R-star aggregators.

One interesting way we can extend these R-star operators is by generalizing the rule which we use to change from one mode of aggregation to the next. Let us consider R_*. For this operator our aggregation rule says

if <u>all</u> the arguments are above e then use the Max <u>else</u> use the Min aggregation.

In the following we use a non-linear function to formally capture this rule. Let Q_* be some function of the arguments defined such that

$$Q_*(x_1, \ldots, x_n) = 1 \qquad \text{if } \underline{\text{at least one}}\ x_i < e$$

$$Q_*(x_1, \ldots, x_n) = 0 \qquad \text{otherwise}$$

We can express the aggregation R_* as

$$R_*(x_1, \ldots x_n) = Q_* \, \text{Min}_i[x_i] + (1 - Q_*) \, \text{Max}_i[x_i]$$

We see that if one element is less than e, then $Q_* = 1$ and $R_* = \text{Min}_i[x_i]$ while if all elements are above e then $R_* = \text{Max}_i[x_i]$.

Now consider another function Q* defined such that

$$Q^*(x_1, \ldots x_n) = 1 \qquad \text{if } \underline{\text{all}}\ x_i < e$$

$$Q^*(x_1, \ldots x_n) = 0 \qquad \text{otherwise}$$

Using Q* we can express

$$R^*(x_1, \ldots x_n) = Q^* \, \text{Min}_i[x_i] + (1 - Q^*) \, \text{Max}_i[x_i]$$

Here we see that if at least one element is less than e then $R^* = \text{Max}_i[x_i]$; otherwise, $R^* = \text{Min}_i[x_i]$.

It is apparent, then, that we can express both R-star operators using some non-linear function Q of the arguments as

$$\text{R-star}(x_i) = Q \, \text{Min}_i[x_i] + (1 - Q) \, \text{Max}_i[x_i].$$

Let us now generalize this form of aggregation. Assume our argument is $X = [x_1, \ldots x_n]$. Let n_T be the number of elements in X that are not equal to e, we will call these "decision elements" and let n_L be the number of elements in X that are less than e. We now define a function Q as follows

$$Q(X) = 1 \qquad \text{if } \frac{n_L}{n_T} > a$$

$$Q(X) = 0 \qquad \text{otherwise}$$

where $a \in [0, 1]$. Figure 2.5 illustrates the definition of Q.

Consider the operator

$$R(x_1, \ldots x_n) = Q(\frac{n_L}{n_T}) \, \text{Min}_i[x_i] + (1 - Q(\frac{n_L}{n_T})) \, \text{Max}_i[x_i]$$

Let us see how this operator works. Obviously $\frac{n_L}{n_T}$ is the proportion of the decision elements that

are below e. If this proportion is greater than a then R implements a Min aggregation; if it is less than or equal to a then R implements a Max operation. If a = 0, then the occurrence of any element less than e causes a Min aggregation, while if there is no such element we get a Max aggregation. This is essentially R_*. If a $\rightarrow$ 1 then we see that we perform Min only if all the elements are below e, otherwise we get a Max. This is essentially R*. We see that as we move from 0 to 1 this changes.

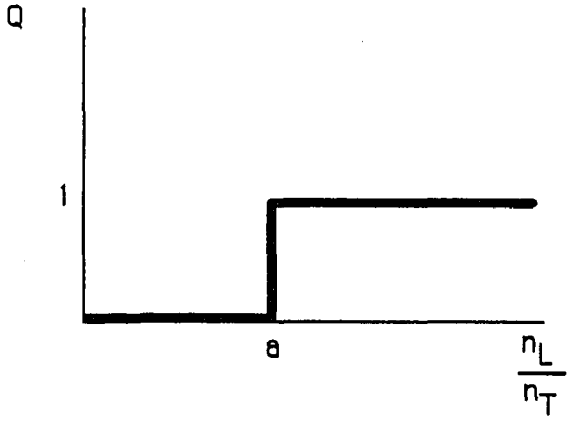

Figure 2.5. Definition of the function Q

Example: Assume we desire to aggregate

$$.[.7,\ 1,\ .5,\ .6,\ .3,\ .2]$$

where e = .5. In this case $n_T = 5$ and $n_L = 2$. Thus $\frac{n_L}{n_T} = \frac{2}{5} = .4$.

(1) a $\rightarrow$ 0. This means if any element is less than .5 we use Min. $\frac{n_L}{n_T} > 0$,

hence $R(x_1, \ldots, x_n) = \text{Min}(x_1, \ldots, x_n) = .2$

(2) a > .4. In this case, $\frac{n_L}{n_T} < a$ and hence we use the Max

$$R(x_1, \ldots, x_n) = \text{Max}(x_1, \ldots, x_n) = 1$$

Let us look at some properties associated with this aggregation. We shall define

$$R_{Q\ star}(x_1, \ldots, x_n) = Q(X)\ \text{Min}_i[x_i] + (1 - Q(X))\ \text{Max}_i[x_i]$$

where

$$Q(x_1, \ldots, x_n) = f(\frac{n_L}{n_T})$$

with

$$f(z) = 0 \qquad \text{if } z \leq a$$

$$f(z) = 1 \qquad \text{if } z > a$$

We first note that $R_{Q\ star}$ is a symmetric/commutative operator, so the indexing of the arguments doesn't effect the aggregation.

Theorem 2.9: $R_{Q\ star}$ is monotone in its argument.

Proof: Without loss of generality we shall consider the case of x_1, that is, we shall show that if x_1 increases $R_{Q\ star}$ can't decrease. We first note that both Max and Min are monotone. Thus if the change in x_1 doesn't cause a change in Q then $R_{Q\ star}$ is monotone. Furthermore, since $Max_i[x_i] \geq Min_i[x_i]$, any change that causes Q to go from 1 to 0 also causes $R_{Q\ star}$ to increase. The only dangerous situation is a case in which an increase in x_1 can cause a change of Q from zero to 1. Assume $Q(x_1, \ldots, x_n) = 0$; this requires that $\frac{n_L}{n_T} < a$. However, an increase in x_1 can't cause $\frac{n_L}{n_T}$ to increase and thus Q can't change from 0 to 1 with an increase in x_1.

Theorem 2.10: $R_{Q\ star}$ has e as an identity.

Proof: Consider the addition of an element of value e to our aggregation. Such an element cannot change $\frac{n_L}{n_T}$. If $\frac{n_L}{n_T} > a$ then Q = 1 and our aggregation is Min, but since at least one element is less than e the addition of e doesn't effect our aggregation. Assume $\frac{n_L}{n_T} < a$. In this case our aggregation is Max and at least one element is greater than e, and thus the inclusion of e doesn't effect our aggregation.

One special case worth noting is when all the elements are e. However we define $\frac{0}{0}$, we must get either Max or Min. Since when all elements are equal to e the idempotency property of Max and Min yield equivalent results, it is the case that if all elements are e then we get e.

Thus we see that $R_{Q\ star}$ has the properties of an MICA operator, i.e., it is commutative, monotone and has a fixed identity.

In the preceding we have suggested that our aggregation $R_{Q\ star}$ be defined as

$$R_{Q\ star}(X) = Q(X)\ Min(X) + (1 - Q(X))\ Max(X)$$

In this formulation the value of Q(X) can be interpreted as the degree of truth of the proposition

"the proportion $\frac{n_L}{n_T}$ is *at least a*",

where a is some value contained in [0, 1]. In the above we can represent *at least a* as a subset, Q, of the unit interval. Then Q(X) is essentially $Q(\frac{n_L}{n_T})$, the membership grade of $\frac{n_L}{n_T}$ in the fuzzy

subset Q.

The concept *at least a* is expressed as a crisp subset of the unit interval. More generally we can use the statement

"the proportion $\frac{n_L}{n_T}$ is Q"

where Q is a linguistic quantifier [Zadeh, 1983] such is *large* or *small*. In particular Q must be a monotone regular quantifier, that is,a fuzzy subset of the unit interval with

(1) $Q(0) = 0$

(2) $Q(1) = 1$

(3) $Q(r) \geq Q(r')$ for $r > r'$

A prototypical example of this is the quantifier *most*, which if used would yield the rule

if most of the decision elements are below the identity then use Min else use Max.

In this situation, if we define the membership function of the fuzzy subset *most* as

$$most(y) = y^2$$

then we get as an aggregation operator

$$R_{\text{most star}} = (\frac{n_L}{n_T})^2 \text{ Min}(x_i) + (1 - (\frac{n_L}{n_T})^2) \text{ Max}(x_i).$$

In this case, when we use fuzzy quantifiers, rather than getting a discrete choice between Max and Min we get some kind of weighted average.

A simpler situation would be if

$$Q(y) = y.$$

In this case we get

$$R_{\text{linear star}} = (\frac{n_L}{n_T})\text{Min}[x_i] + (\frac{n_T - n_L}{n_T}) \text{ Max}[x_i]$$

Thus we see that the process of selection between Max and Min is based upon some fuzzy rule and here we take some combination. Thus we have suggested

$$R_{Q\text{-star}}(X) = Q(\frac{n_L}{n_T}) * \text{Min}(X) + (1 - Q(\frac{n_L}{n_T})) \text{ Max}(X)$$

In the above $Q(\frac{n_L}{n_T})$ is the firing condition specified in terms of the proportion of below e elements.

There exists another type of generalization which we shall briefly touch upon. Assume we divide the collection of arguments, X, into three categories:

H: arguments above e

L: arguments below e

M: arguments equal to e

We note n_L = card(L) and n_T = card(L) + card(H). Let T and S be any t-norm and t-conorm respectively. We can define

$$R_{Q\ star}(X) = Q(\frac{n_L}{n_T})\ T(L) + (1 - Q(\frac{n_L}{n_T}))\ S(H).$$

It is obvious that this operation is independent of the indexing of the x_i, and therefore commutative. It has e as an identity since none of the terms are effected by the addition of an argument equal to e. Finally, this aggregation is monotone since the increase of any element can't cause a decrease in R. Thus this has the properties of a MICA operator.

3. Full reinforcment operators in Aggregation Techniques.

3.1. Reinforcement in Aggregation.

In pattern recognition techniques we generally have a set of criteria or characteristics used to help in identifing an object as belonging to some class. Typically in these procedures an unclassified object is scored as to how well it satisfies each of the criteria. Unless stated otherwise we shall assume this scoring takes place on the unit interval. These scores are then aggregated to obtain some overall value indicating how well the unclassified objects satisfy the condition of this class . A fundamental aspect of this procedure is the aggregation step. Two properties generally required of such an aggregation procedure are commutativity and monotonicity. By commutativity we mean that the order in which we index the characteristics used to identify the object doesn't effect the aggregation. By monotonicity we mean that as the scores of the individual criteria increases the overall score increases.

In [Elkan ,1993] Elkan pointed out another property that is often used by human reasoners in pattern recognition type problems and may be a desirable property to include in some automated systems, particularly if we desire to call them intelligent. We call this property **full reinforcement**. In order to introduce this concept we must introduce two closely related ideas. The first concept is called **upward reinforcement**. Informally by upward reinforcement we mean to indicate the situation where if the characteristics used to identify an object are all highly satisfied then these scores reinforce each other to give us an even higher confidence that the objective belongs to the class (upward reinforcement is an often observed phenomena in human reasoning). Similarly by **downward reinforcement** we mean to conceptualize the observation that if the scores are all low then these scores reinforce each other to give us an even greater confidence that the object doesn't belong to the class. We shall call an aggregation operator displaying both upward and downward reinforcement a **full reinforcement** operator.

In order to formally investigate the property of full reinforcement we shall make some

tentative definitions for upward and downward reinforcement. An aggregation operator M is said to display upward reinforcement if it is always the case that when all the characteristics are affirmative

$$M(a_1, ..., a_n) \geq Max_i[M(a_i)].$$

Similarly the operator M is said to display downward reinforcement if it is always the case that when all the characteristic are disaffirmative

$$M(a_1, ..., a_n) \leq Min_i[M(a_i)].$$

Example: Assume a class of objects are characterized by three properties, A_1, A_2, A_3. Let M be an aggregation operator such that M(a) = a. If for object X we have $A_1(x) = 0.9$, $A_2(x) = 0.91$ and $A_3(x) = 0.88$ and if

$$M(0.9, 0.91, 0.88) = 0.97$$

then M is an upward reinforcement operator.

If we for object y we have $A_1(y) = 0.2$, $A_2(y) = 0.25$, $A_3(y) = 0.18$

and if

$$M(0.2, 0.25, 0.18) = 0.1$$

then M is a downward reinforcement operator.

Since many aspects of human reasoning exhibit this full reinforcement property it appears to be a desirable quality to have in some types of aggregation operators. It is our purpose here to investigate classes of aggregation operators which manifest this property.

We began our search by considering some established classes of aggregation operators and see if these do indeed provide full reinforcement. The t-norm operator [Alsina, Trillas, Valverde, 1983], [Weber, 1983], [Klement, 1982] has found application in the fuzzy set literature as a generalization of the conjunction, *anding* operation. Three prototypical examples of this operator are

1. T(a, b) = Min (a, b)
2. T(a, b) = a b
3. T(a, b) = Max(0, a + b - 1)

We should note that the associativity property allows us to naturally extend a given t-norm to any number of arguments.

A fundamental property associated with the t-norm is

$$T(a_1,.....a_n) \leq Min_i[a_i] \qquad I.$$

From this property we can see that the t-norm type aggregation exhibits the property of downward reinforcement. In particular we see that the aggregation is always less then the minimum of the arguments. An alternative view of this downward reinforcement can be seen from the condition that

$$T(a_1, a_2,...a_n) \geq T(a_1, a_2, ..., a_{n+1}) \qquad II.$$

Here we see that if $a_1.....a_n$ are low scores then the addition of another low score will even further decrease the aggregation value.

Unfortunately the t-norm operator doesn't exhibit upward reinforcement. Consider the case in which

$$a_1 = .95, a_2 = .97, a_3 = .98$$

if we have

$$T(a, b) = a\ b$$

then

$$T(a_1, a_2) = .9215$$

and

$$T(a_1, a_2, a_3) = .903$$

Thus the inclusion of a very high score doesn't reinforce the other scores. The lack of upward reinforcement is implicit in the conditions I and II held by any t-norm. Based upon these observations we see that the t-norm operator is not a full reinforcement operator.

We now turn to the t-conorm aggregation operator [Alsina, Trillas, Valverde, 1983], [Weber, 1983], [Klement, 1982]. This operator has found use in the fuzzy set literature as a generalization of disjunction, *or* type aggregation. Three prototypical examples of this operator as

1. $S(a, b) = Max(a, b)$
2. $S(a, b) = a + b - a\ b$
3. $S(a, b) = Min[1, a + b]$

A fundamental property associated with the t-conorm is that

$$S(a_1....a_n) \geq Max_i[a_i].$$

From this property we that the t-conorm is an upward reinforcement operator, for we see that when all a_i's are high the aggregated value will be at least as big as the largest value. On the other hand we see that these operators are not downward reinforcement operators. In the case when all scores are low this doesn't aggregate to a score less than the Min but rather to one at least as great as the Max of the aggregated values.

In the proceeding we have indicated that neither the t-norm nor the t-conorm operators are full reinforcement operators. We have shown that each are one sided reinforcement operators, the t–norm being downward reinforcing while the t-conorm is upward reinforcing. From this observation it appears natural to try to use some linear combination of t-norm and t-conorm to construct a full reinforcement operator. As the following counter-example shows, this task is impossible.

Example: Assume T and S are some arbitrary t-norm and t-conorm. Let $\alpha \in [0,1]$ and consider the operator

$$G(a, b) = \alpha\, T(a, b) + (1-\alpha)\, S(a, b).$$

Consider the input

$$a = .95 \text{ and } b = .98$$

For G to be upward reinforcing we require

$$\alpha\, T(.95, .98) + (1-\alpha)\, S(.95,.98) \geq .98.$$

Since

$$S(.95, .98) \leq 1$$

$$T(.95,.98) \leq .95$$

then upward reinforcement can be satisfied only if

$$\alpha\, (.95) + (1-\alpha)\, 1 \geq .98.$$

From this we see that it is required that

$$\alpha \leq 2/5.$$

Now consider the input

$$a = .05 \text{ and } b = .02$$

For G to be a downward reinforcing operator we require that

$$\alpha\ T(.05, .02) + (1-\alpha)\ S(.05, .02) \leq .02$$

Since

$$T(.05, .02) \geq 0$$

$$S(.05, .02) \geq .05$$

then downward reinforcement can be satisfied only if

$$\alpha\ (0) + (1-\alpha)\ .05) \leq .02$$

From this we see that we require

$$\alpha \geq 3/5.$$

Finally we see that the two requirements, $\alpha \geq 3/5$ and $\alpha \leq 2/5$ are contradictory and hence we can't satisfy both upward and downward reinforcement by a linear combination of any t-norm and t–conorm based on a fixed value α.

We now examine the class of mean operators [Dubois, Prade,1985], [Yager,1993] to see if they are full reinforcement operators. An mean operator A is a mapping

$$A:\ I^n \rightarrow I$$

such that it is

(1) Monotonic

(2) Commutative

(3) Idempotent $A(a,...., a) = a.$

It can be shown that a fundamental property [Dubois, Prade,1985], [Yager,1993] associated with mean operators is that

$$Min_i[a_i] \leq A(a_1...., a_n) \leq Max_i\ [a_i].$$

From this property we see that a mean operator is neither an upward nor downward reinforcement operator and thus is not a full reinforcement operator. For we see that even when scores are high the aggregated value is still bounded above by the Max of the arguments. Similarly when all scores are low it is bounded below by the Min of the arguments. As a matter of fact we see that the mean operator exhibits a property, which we call a centering property [Yager, 1993], which is antithetical to the reinforcement property.

We have shown that the classic *and* and *or* operators, and any linear combination of these, as well as the mean operator don't display the property of full reinforcement. We must now search elsewhere to find the necessary operators to provide us with the desired full reinforcement.

3.2. MICA Operators and Reinforcement

In the section we discuss a class of aggregation operators called MICA operators [Yager,1994]. We discuss a few properties of these operators and describe some subclasses of these operators. We show that their exists a significant subclass which has the desired full reinforcement property.

Assume I is the unit interval. A bag drawn from I is any collection of elements all of which are contained in I. A bag is different from a subset in that it allows multiple copies of the same element. A bag is similar to a subset in that the ordering of the elements doesn't matter. We shall use the notation $< a_1, ..., a_n>$ to indicate a bag. In the following we shall let U^I indicate the subset of all bags of the set I. If A and B are two bags drawn from I we shall let $A \oplus B$ indicate the new bag consisting of all elements in A and B.

Definition 3.1: Assume A and B are two bags of the same cardinality. If the elements in A and B can be indexed in such a way that $a_i \geq b_i$ for all i then we shall denote this as $A \geq B$.

A function F: $U^I \rightarrow I$ is called a **bag mapping** from U^I into the unit interval. An important property of bag mappings is that they are commutative in the sense that the ordering of the elements doesn't matter. Thus if $Agg(x_1, ..., x_n)$ is an aggregation operator representable by a bag mapping then Agg must be a commutative type aggregation.

Definition 3.2: Assume F is a bag mapping from U^I into I and let A be any bag. An element $u \in I$ is called an **identity** for A under F if

$$F(A) = F(A \oplus <u>)$$

Essentially we see that an identity is some element which can be added to the bag A without changing its evaluation under the mapping F.

Bag mappings allows us to formulate aggregation operators in which we can represent the

arguments of the aggregation by a bag B. In the following we introduce a class of bag mappings called MICA operators [Yager,1994]. In [Yager,1994] Yager showed their central rule in the construction of fuzzy system models.

Definition 3.3: A bag mapping M,

$$M: U^I \rightarrow I$$

is called a MICA (Monotonic Identity Commutative Aggregation) operator if it has the following two properties

1) Monotonicity: If $A \geq B$ then $M(A) \geq M(B)$.

2) Identity: For every bag A there exists an element, $u \in I$ called the identity of A under M such that if $D = A \oplus \langle u \rangle$ then $M(D) = M(A)$.

We note that the inherent commutativity of any bag mapping assures us of the commutativity of the MICA operators.

We shall say that MICA operator M has **natural boundaries** if for any $x \in I$

$$M(\langle x \rangle) = x.$$

Thus natural boundaries implies that the evaluation of any one element bag is the value of the element in the bag. When used as aggregation operators naturally bounded MICA operators imply that the degree of satisfaction of any one element is value of that element. It should be carefully pointed out that in general MICA operators are <u>not</u> necessarily naturally bounded.

The type of identity element associated with a MICA operator provides a very useful way of classifying these types of operators. Two important classes of these MICA operators have been identified [Yager,1994].

Definition 3.4: A MICA operator M is said to have a **fixed identity** if there exists some element $g \in I$ such that

$$M(A) = M(A + \langle g \rangle)$$

for all A.

Thus the fixed identity MICA operators have the same identity for all bags and the identity is solely dependent on the operator M.

Definition 3.5: A MICA operator M is said to have a **self identity** if for every bag A the identity element u is such that u = M(A).

For a self identity MICA operator the identity is the value of evaluation of A and changes with different A's.

Let us now look at the class of fixed identity MICA operators which we shall denote as **FIMICA** operators. Assume D is a any bag and let M be any FIMICA operator with identity g. We can always express D as

$$D = A \oplus G$$

where A is the subbag of D consisting of those elements that are not equal to g and G is the subbag of g values in D. It can be easily shown that

$$M(D) = M(A).$$

Thus we can always discard any scores equal to the identity. If Φ is the empty bag it can easily be shown that

$$M(\Phi) = M(<g>).$$

We see this as follows

$$M(<g>) = M(\Phi \oplus <g>) = M(\Phi).$$

A FIMICA operator will be called **normal** if

$$M(\Phi) = g.$$

Thus we see that for normal MICA operators

$$M(<g>) = g.$$

The following theorem provides a fundamental property of the class of FIMICA operators

Theorem 3.1: Assume M is a FIMICA operator with fixed identity g, let A be any bag drawn from U^I and let $\beta \in I$. Then it is the case that

$$M(A \oplus <\beta>) \geq M(A) \quad \text{if } g \leq \beta$$

$$M(A \oplus <\beta>) \leq M(A) \quad \text{if } g \geq \beta$$

Proof: Let $A = < a_1, \ldots, a_n>$ and let $B = <b_1, \ldots, b_n, b_{n+1}>$ where $b_i = a_i$ for i = 1 to n and $b_{n+1} = \beta$. Let $B' = <b_1, \ldots, b_n, g>$. From the identity property of M we have

$$M(B') = M(A).$$

If $\beta > g$ then $B > B'$ and from monotonicity we get

$$M(B) \geq M(A)$$

If $\beta < g$ then $B < B'$ and we get from monotonicity

$$M(B) \leq M(A)$$

Thus we see that the fixed identity provides a special element such that adding elements above this value tends to increase the value of the aggregation, can't decrease it, while adding values below this value tends to decrease the value of the aggregation, can't increase it.

What is important to observe is that having chosen g allows us to define affirmative or good scores, those scores above g, and disaffirmative, bad scores, those scores below g.

The following theorem shows how these ideas lead to a form of full reinforcement in FIMICA operators.

Theorem 3.2: Assume M is a FIMICA operator with identity g. Let $A = \langle a_1, ..., a_n \rangle$ be any bag such that $a_i > g$ for all a_i in A. Then

$$M(A) \geq Max_i[M(\langle a_i \rangle)].$$

Proof: Let us denote A_i as the bag of dimension n consisting of the element a_i and having all other elements equal to g. From the fixed identity property it follows

$$M(A_i) = M(\langle a_i \rangle) \text{ for all } i$$

and hence

$$M(A) \geq Max_i[M(\langle a_i \rangle)]$$

Thus we see that if we define the concept of affirmative scores as being those greater than the g the FIMICA operator exhibits upward reinforcement as described earlier.

As the following theorem shows if we define the concept of disaffirmative scores as being those less than g the FIMICA operator also exhibits downward reinforcement

Theorem 3.3: Assume M is a FIMICA operator with identity g. Let $B = \langle b_1, ..., b_n \rangle$ be any bag such that $b_i < g$ for all b_i in B. Then it is the case that

$$M(B) \leq Min_i[M(\langle b_i \rangle)]$$

Thus we see that the class of FIMICA operator can exhibit both upward and downward

reinforcement and thus are full reinforcement operators.

As we showed the FIMICA operators exhibit both types of reinforcement because there exists some fixed element g for which elements above g cause an increase in the score while those below g cause a decrease in the score. In this regard the following observation is worth noting. Assume $g = 0$, in this case all elements in any bag are such that $a_i \geq g$ and thus for any bag A, $M(A) \geq Max_i[M(<a_i>)]$. In particular no possibility exists for downward reinforcement. Thus in order to have the possibility for downward reinforcement we need to have $g > 0$. Similarity if $g = 1$ then for all elements in any bag $a_i \leq g$ and it is the case that $M(A) \leq Min_i[M(<a_i>)$ and no possibility exists for upward reinforcement. Thus we see that for a MICA operator to have the possibility to exhibit both kinds of reinforcement g must be neither 1 or 0. Thus only the situation of $0 < g < 1$ allows for both kinds of reinforcement.

Assume D is a bag and M is a FIMICA operator with identity g. Let us express

$$D = A \oplus B \oplus G$$

where A consists of all the elements in D greater the g, B consists of all the elements of D less the g and G consists of all the elements equal g. As we have shown

$$M(A) \geq Max_i[M(<a_i>)]$$

$$M(B) \leq Min_i[M(<b_i>)]$$

Consider the bag $D' = A \oplus B' \oplus G$ where B' is a bag of the same cardinality as B but with all the elements equal to g. Since all the elements in B are less then g and thus

$$D' > D$$

Furthermore since

$$M(D') = M(A)$$

we get form monotonicity that

$$M(D) \leq M(A)$$

similarly if $D'' = A' \oplus B \oplus G$ where A' is a bag of the same cardinality as A with all the elements equal to g we can show

$$D'' < D$$

and have

$$M(D) \geq M(B)$$

Thus we see that

$$M(B) \leq M(D) \leq M(A).$$

In the preceding we have shown that the class of FIMICA operators with fixed identity $0 < g < 1$ provide a class of full reinforcement operators.

We now turn to look at the potential for self identity type MICA operators to provide for full reinforcement type operators. Our first result is not very optimistic regarding the goal.

Theorem 3.4: Any self identity MICA operator which is naturally bounded is a mean operator.

Proof: Since all MICA operators are commutative and monotonic all we need to prove is idempotency which can be easily shown as follows. Natural boundedness assures us that

$$M(<a>) = a$$

and therefore

$$M(<a, a>) = a.$$

Adding a to <a, a> also gives its an evaluation of a. Continuing in this manner we get idempotency.

Since we have shown that the prototypical mean operator is not reinforcing we are not optimistic. More generally we see that, because of idempotency, a collection of equal scores must evaluate to the same score so that we can't get any reinforcement.

Even if we don't assume natural boundaries we see that self identity type MICA operators are not good for reinforcement. If M is a self identity operator then Yager [7] has shown the following property holds for any bag A and element β

$$M(A \oplus <\beta>) \geq M(A) \qquad \text{if } \beta > M(A)$$

$$M(A \oplus <\beta>) \leq M(A) \qquad \text{If } \beta < M(A)$$

The important observation is that the ability of an element β to provide upward or downward reinforcement depends upon the value M(A), in essence the concept of a good score becomes a moving target which is dependent on the other scores. Assume the elements of A are all high (above .91) and thus results in a high $M(A) > 0.91$. Consider the addition of $\beta = .9$ even though this is a high value it still won't cause any reinforcement because it is less then the value of M(A). Thus we see that self identity operators, which are essentially mean type aggregators, don't provide

a fertile class for looking for full reinforcement operators. Because of this we shall concentrate only on FIMICA operators.

We now turn to the issue of aggregation of weighted bags under FIMICA operators. A weighted bag on I is a bag whose elements are tuples (ω_i, α_i). For any tuple ω_i is the weight (importance or relevance) associated with an argument, and α_i is the value of the argument. In the following we assume both ω_i and α_i are drawn from I. The process of evaluating (aggregating) the elements in a weighted bag consists of two steps. The first step is to convert the elements in the weighted bag, the tuples, into single values called effective values. The second step consists of aggregating the effective values using the FIMICA operator as previously described.

The operation of changing the tuples into effective values is called the transformation process. In this process the elements in the original weighted bag A, (ω_i, α_i), are transformed into effective values, b_i, which gives us a simple bag B of the same cardinality as A.

In [Yager,1994] Yager discussed an operator which can be used to do this transformation,

$$h(\omega, \alpha) = b$$

In constructing such a transformation operator we desire it to have at least the following four properties:

(1) Monotonicity with respect to the value, α. In particular if $\alpha > \alpha'$ then we require

$$h(\omega, \alpha) \geq h(\omega, \alpha').$$

(2) A normalcy with respect to the weights,

$$h(1, \alpha) = \alpha.$$

(3) We desire that elements with weight zero have no effect on the aggregation process, thus if g is the fixed identity of the aggregation to be used on the resulting bag we must have

$$h(0, \alpha) = g.$$

(4) Finally we desire that the transformation moves monotonically from its value for $\omega = 0$ to $\omega = 1$. In particular

$$\frac{\partial}{\partial \omega} h(\omega, \alpha) \geq 0 \qquad \text{if } \alpha \geq g$$

$$\frac{\partial}{\partial \omega} h(\omega,\alpha) \leq 0 \qquad \text{if } \alpha \leq g$$

We call this the consistency with respect to the importance property.

One formulation that satisfies these conditions is

$$h(\omega, b) = \omega\, b + \overline{\omega}\, g$$

where $\overline{\omega} = 1 - \omega$. We can express this alternatively as

$$h(\omega, b) = \omega(b - g) + g.$$

We note in the special case when $g = 0$ we get $h(\omega, b) = \omega\, b$ and in the special case when $g = 1$ we get $h(\omega, b) = \omega\, b + \overline{\omega}$.

In general the form for h depends upon the identity of the FIMICA operator which is to be used to carry out the aggregation . This requirement is implicit in the third property associated with h.

3.3. Some Families Of FIMICA Operators

In the proceeding we have shown that FIMICA operators, with $0 < g < 1$, provide a class of potential full reinforcement operators. We now provide some families of this type of operator.

The first family of FIMICA operators we investigate are an additive family of aggregation operators. In the following we shall assume f is a mapping from the real numbers to the unit interval,

$$f: R \rightarrow I.$$

In addition we shall assume that f is monotonic

$$f(x) \geq f(y) \qquad \text{if } x \geq y.$$

Consider now the aggregation operator

$$M(A) = f(\sum_{i=1}^{n} (a_i - g))$$

where $g \in [0, 1]$ and A is any bag, $A = \langle a_1, \ldots, a_n \rangle$. It is obvious that M is commutative and monotonic with respect to the arguments in A. Since $a_i - g = 0$ if $a_i = g$ then the addition of an

element with value g doesn't change the value of M(A) and hence g is a fixed identity. Thus the above operator is a FIMICA. It should be pointed out that this additive family of FIMICA operators is not associative. It can be seen to be quasi- associative in that the argument of f is associative.

Normalcy of these operators can be had if we further assume that

$$f(0) = g.$$

In using this class of operator the function f essentially determines the "severity" of the aggregation process being carried out. The actual form of f will be problem dependent and in some cases may be obtained by some learning algorithm

A further generalization can be made by considering the form

$$M(A) = f\left(\sum_{i=1}^{n} P_g(a_i)\right)$$

where

$$P_g: I \rightarrow R$$

such that

(1) $P_g(g) = 0$

(2) P_g is monotonic $P_g(x) \geq P_g(y)$ when $x \geq y$

Actually we can have a different P_g for each element i.

We now consider the case where our elements come from a weighted bag. We see that each element a_i is first transformed into b_i where $b_i = h(\omega_i, a_i)$. In the case of h previously introduced we get

$$b_i = \omega_i a_i + \bar{\omega}_i g.$$

For this situation we get

$$M(A) = f(\textstyle\sum_i (b_i - g))$$

$$M(A) = f(\textstyle\sum_i (\omega_i a_i + \bar{\omega}_i g - g))$$

$$M(A) = f(\textstyle\sum_i (\omega_i a_i - \omega_i g))$$

$$M(A) = f(\textstyle\sum_i (\omega_i (a_i - g)))$$

A special case of this occurs when g = 0 in this case

$$M(A) = f(\textstyle\sum_i (\omega_i a_i)).$$

This is the type of aggregation that is typically used at the nodes in neural networks. However, as we noted the use of g =0 only allows upward reinforcement.

If we return to our unweighted form

$$M(A) = f(\sum_i (a_i - g))$$

we can consider a quasi-linear form for f

$$f(x) = 0 \qquad x \le g/K$$

$$f(x) = Kx + g \qquad g/K \le x \le 1 - g/K$$

$$f(x) = 1 \qquad x \ge 1 - g/K$$

where $K \ge 0$. If $g = 0$ and $K = 1$ this becomes the so called bounded sum aggregation [10].

A closely related family of FIMICA operators can be had if we consider a mapping

$$\hat{f}: R \rightarrow I$$

where $\hat{f}$ is anti-monotone

$$\hat{f}(x) \le \hat{f}(y) \qquad \text{if } x \ge y$$

In this case it can be shown that

$$M(A) = \hat{f}(\sum_{i=1}^{n} (g - a_i))$$

satisfies the required conditions of a FIMICA operator.

If we assume the elements are weighted we get

$$M(A) = \hat{f}(\sum_{i=1}^{n} (g - (\omega_i a_i + \overline{\omega}_i g)))$$

$$M(A) = \hat{f}(\sum_{i=1}^{n} (\omega_i (g - a_i))).$$

We now turn to another class of FIMICA operators. We call this the product type operators. Consider the aggregation

$$M(A) = f(\prod_{i=1}^{n} \frac{a_i}{g})$$

where $g > 0$. We assume f is a mapping

$$f: R \rightarrow I$$

such that f is monotone.

We easily see that M(A) is monotone and commutative. In addition since for $a_i = g$, we get

$\frac{a_i}{g} = 1$ then g is indeed an identity. Thus the above M(A) is a FIMICA operator for $g > 0$. This aggregation has one notable characteristic, if $a_i = 0$ for any i it always gives the value f(0). Essentially this property models a situation where the occurrence of an argument of 0 imposes the restriction that the value of the aggregation will be the lowest possible value. It should be noted that the t-norm operators also have this property. A closely related class of aggregation operators is

$$M(A) = f\left(\prod_{i=1}^{n} \frac{K + a_i}{K + g}\right)$$

where $K > 0$. First we see it is definable for all g. Secondly it can be easily be shown to be a FIMICA operator. Furthermore it doesn't become fixed at any value when $a_i = 0$ for some i.

3.4. Uni-norm Aggregation Operators as Reinforcement Operators

Earlier in the paper we introduced the concept of the t-norm and the t-conorm operators. These operators are actually special cases of FIMICA operators. The monotonicity and commutativity are guaranteed by their definitions. For the t-norm it is easily shown that one is the identity value and for the t-conorm the zero is the identity. This observation now makes clear the reason why these operators are not full reinforcement operators. As we have shown any full reinforcement requires that the identity be neither one nor zero. Since the t-norm and t-conorm have one and zero respectively as their identities they couldn't be full reinforcement operators. While these operators are not full reinforcement operators they have one nice property, that of associativity, which would be desirable in an aggregation operator. Associativity brings with it two important benefits. The first is the ability to easily calculate the aggregated value as we add elements. The second and perhaps more important is that once having defined the aggregation operation for all pairs of two elements we have defined it for any number of elements.

The uni-norm operators satisfies the first three of these properties of t-norm and t–conorm but the fourth condition is more general in that it allows for any identity. Using the associativity property we can extend R so that it works for arguments $n \geq 2$. Thus if A is any bag of dimension $n \geq 2$ we shall denote this as R(A) and readily evaluate.

Using the structure of a uni-norm we can introduce a class of bag mappings.

Definition 3.6: *Assume R is an uni-norm with identity g. A bag mapping M is called* **a uni-norm bag mapping** based on R if

(1) For the null bag

$$M(\Phi) = g.$$

(2) For any singleton bag

$$M(<u>) = u.$$

(3) For any bag A of cardinality $n \geq 2$

$$M(A) = R(A)^{1}.$$

The following theorem shows that any uni-norm bag mapping is a FIMICA mapping.

Theorem 3.5: Any uni-norm bag mapping M based upon a uni-norm R is a FIMICA operator with identity equal to the identity of R.

Proof: (1) Commutativity follows directly from the commutativity of R.

(2) Monotonicity:

i) If A has dimension one then since $M(<u>) = u$ we have monotonicity.

ii) If $\dim(A) \geq 2$ then monotonicity follows from the monotonicity of R.

(3) Fixed identity requires that for any bag A, $M(A) = M(A \oplus <g>)$

i) Assume $A = \Phi$. Consider

$$B = \Phi \oplus <g> = <g>.$$

Then

$$M(\Phi) = g$$

$$M(\Phi + <g>) = M(<g>) = g$$

[1]We note that R is an associative operator thus we can evaluate, using the assocativity property, R for any any bag of dimension $n \geq 2$.

Thus the condition is satisfied

ii) Assume A is a bag consisting of one element, A = <x>, in this case M(<x>) = x. Consider now the bag <x, g>. From the definition of R we have M(<x, g>) = R(x, g) = x. Thus

$$M(<x>) = M(<x> \oplus <g>).$$

iii) Assume A is any bag of dimension $n \geq 2$. From the associativity property we get that $M(A \oplus <x>) = R(M(A), x)$ and hence if x = g we get

$$M(A \oplus <x>) = R(M(A), g) = R(M(A)) = M(A).$$

As a result of this theorem we see that the uni-norm bag mappings are FIMICA operators which have two special properties

(1) natural boundaries: M(<x>) = x

(2) Associativity

As discussed earlier the fact that they are FIMICA operators means that they can be used as full reinforcement operators when g is not 1 or 0.

Having introduced the concept of uni-norm type FIMICA operators, we next turn to the question of the existence or construction of such operators. The construction of these operators depends, of course, on the existence of uni-norms. We first note that for g = 1 or 0 there exists a large class of such uni-norms corresponding to the t-norms and t-conorms respectively. However, for the purpose of finding full reinforcement operators we need uni-norms with identity other then 1 or 0. In [Fodor, Yager, Rybalov, 1994] Fodor, Yager and Rybalov introduced a general class of uni-norms for any g. In the following we describe this class.

Let T and S be any t-norm and t-conorm respectively. The mappings R defined by **I** and **II** below are two general classes of uni-norm operators with identity g.

I.

(a) $R(x, y) = g\, T(\frac{x}{g}, \frac{y}{g})$ if $0 \leq x, y \leq g$

(b) $R(x, y) = g + (1 - g)\, S(\frac{x - g}{1-g}, \frac{y - g}{1-g})$ if $g \leq x, y \leq 1$

(c) $R(x, y) = \text{Min}(x, y)$ if $\text{Min}(x, y) \leq g \leq \text{Max}(x, y)$

II.

(a) $R(x, y) = g\, T(\frac{x}{g}, \frac{y}{g})$ if $0 \le x, y \le g$

(b) $R(x, y) = g + (1 - g)\, S(\frac{x - g}{1-g}, \frac{y - g}{1-g})$ if $g \le x, y \le 1$

(c) $R(x, y) = \text{Max}(x, y)$ if $\text{Min}(x, y) \le g \le \text{Max}(x, y)$

The difference between the two classes are in item c. In particular we note that for the first class $R(0, 1) = 0$ and for the second class $R(0, 1) = 1$.

Let us look at these operators for some special cases of T and S. Assume

$$T(a, b) = ab$$

$$S(a,b) = a + b - ab.$$

In this case

$$g\, T(\frac{x}{g}, \frac{y}{g}) = \frac{xy}{g}$$

$$g + (1 - g)\, S(\frac{x - g}{1-g}, \frac{y - g}{1-g}) = \frac{x + y - xy - g}{1 - g}$$

Now assume

$$T(a, b) = \text{Min}(a, b)$$

$$S(a, b) = \text{Max}(a, b)$$

In this case

$$g\, T(\frac{x}{g}, \frac{y}{g}) = g\,(\frac{x}{g} \wedge \frac{y}{g}) = x \wedge y$$

$$g + (1 - g)\, S(\frac{x - g}{1-g}, \frac{y - g}{1-g}) = x \vee y.$$

Assume

$$T(a, b) = 0 \vee (a + b - 1)$$

$$S(a, b) = 1 \wedge a + b$$

In this case

$$g\, T(\frac{x}{g}, \frac{y}{g}) = 0 \vee (x + y - g)$$

$$g + (1 - g)\, S(\frac{x - g}{1-g}, \frac{y - g}{1-g}) = 1 \wedge (x + y - g)$$

3.5. Reinforcement Aggregation From Fuzzy Models

In this section we look at an alternative approach to the construction of reinforcement operators. This approach is based upon the use of the fuzzy system modeling technique [Yager, Filev, 1994] which has been used with considerable success in construction of fuzzy logic controllers.

First we recall that the t-norm aggregation provides downward reinforcement. On the other hand the t-conorm aggregation provides upward reinforcement. Thus we can see that one way to construct a full reinforcement operator is to have it act like a t-norm when the scores are low and have it act like a t-conorm when the scores are high. We can thus express the basic characteristics of a generic full reinforcement operator in terms of a two rule knowledge base

R1: *If the scores are all low then use a t-norm aggregation*

R2: *If the scores are all high then use a t-conorm aggregation.*

Using these two rules let us construct a fuzzy logic model [Yager, Filev, 1994] and see the type of operator we get using this approach. Assume aggregation consists of n arguments. Let V_j be the variable corresponding to the j^{th} value to be aggregated. Let L_j be the fuzzy subset defined on the unit interval corresponding to the concept *low* for V_j. Let H_j be the fuzzy subset *high* also defined on the unit interval. Furthermore, let T and S be any t-norm and t-conorm respectively. Using these ideas we get a two rule fuzzy model

R1: If V_1 is L_1 and V_2 is L_2....and V_n is L_n then M(A) is B_1.

R2: If V_1 is H_1 and V_2 is H_2....and V_n is H_n then M(A) is B_2.

In the the above B_1 and B_2 are singleton fuzzy subsets

$$B_1 = \left/ \frac{1}{T(V_1, V_2, ..., V_n)} \right.$$

$$B_2 = \left/ \frac{1}{S(V_1, V_2, ..., V_n)} \right..$$

We see that in the above model the consequent value is a function of the input variables, this is a Sugeno-Takagi [Takagi, Sugeno, 1985] type model.

If we assume that our argument is the bag $A = < a_1,..., a_n>$ then we have

$$V_i = a_i$$

as our inputs. As discussed in [Yager, Filev, 1994] the solution of such a fuzzy system model consists of four steps:

(1) Find the firing level τ_i of each rule

(2) Find the effective output fuzzy subset of each rule

(3) Aggregate these effective outputs

(4) Defuzzify the resulting fuzzy set

To find τ_1 we first find the satisfaction τ_{1j} of each antecedent in rule 1, in this case we get

$$\tau_{1j} = L_j(a_j)$$

To obtain τ_1 we can use

$$\tau_1 = \prod_{j=1}^{n} L_j(a_j)$$

or

$$\tau_1 = \text{Min}_j\, L_j(a_j).$$

Similarly for rule two we get

$$\tau_2 = \prod_{j=1}^{n} H_j(a_j)$$

or

$$\tau_2 = \text{Min}_j\, H_j(a_j).$$

Since the consequence of each rule is a singleton we get as the effective output of each rule

$$F_1: \left\{ \frac{\tau_1}{T(a_1, a_2, \ldots, a_n)} \right\}$$

$$F_2: \left\{ \frac{\tau_2}{S(a_1, a_2, \ldots, a_n)} \right\}$$

Aggregating these two rule outputs gives us the system output

$$F = \left\{ \frac{\tau_1}{T(a_1, a_2, \ldots, a_n)}, \frac{\tau_2}{S(a_1, a_2, \ldots, a_n)} \right\}$$

Using the COA method of defuzzification we get

$$M(A) = \frac{\tau_1\, T(a_1, a_2, \ldots, a_n) + \tau_2\, S(a_1, a_2, \ldots, a_n)}{\tau_1 + \tau_2}$$

If we denote

$$\omega_1 = \frac{\tau_1}{\tau_1 + \tau_2}$$

$$\omega_2 = \frac{\tau_2}{\tau_1 + \tau_2}$$

then

$$M(A) = \omega_1 T(a_1, ..., a_n) + \omega_2 S(a_1, ..., a_n).$$

Thus we see that M(A) is obtained as a kind of weighted average of the S and T aggregation of the arguments. We note, however, that this is not a simple weighted average but the weights themselves depend upon the arguments, the elements in the bag A.

If we denote $T(a_1, ..., a_n)$ simply as T and $S(a_1, ..., a_n)$ as S we express this as

$$M(A) = \omega_1 T + \omega_2 S.$$

We should note that this differs from a simple weighted average of a t-norm and t-conorm, which we have previously shown not to exhibit full reinforcement, in that the weights are not constants but depend upon the a_i, hence this is a nonlinear averaging.

We can also express M(A) as

$$M(A) = \omega_1 T (1 + (\frac{\omega_2}{\omega_1} \frac{S}{T})).$$

We shall now consider a special case where we use the product t-norm and its dual for the t conorm, hence

$$T = T(a_1, ..., a_n) = \prod_{j=1}^{n} a_j$$

$$S = S(a_1, ..., a_n) = 1 - \prod_{j=1}^{n} \bar{a}_j$$

If we define the antecedent fuzzy subset *High* using a linear membership function we get

$$H_j(a_j) = a_j.$$

Furthermore if we define *Low* as the negation of *High* then we get

$$L_j(a_j) = 1 - a_j.$$

Using the product form to combine the satisfactions to the individual antecedents we get

$$\tau_2 = \prod_{j=1}^{n} H_j(a_j) = \prod_{j=1}^{n} a_j = T$$

$$\tau_1 = \prod_{j=1}^{n} L_j(a_j) = \prod_{j=1}^{n} \bar{a}_j$$

If we denote $\prod_{j=1}^{n} \bar{a}_j$ as Q we see that S = 1 - Q. Using this notation we get that

$$\frac{\omega_2}{\omega_1} = \frac{\tau_2}{\tau_1} = \frac{T}{Q}$$

Hence we get as our aggregated output

$$M(A) = \omega_1 T (1 + \frac{T}{Q} \frac{1 - Q}{T})$$

$$M(A) = \omega_1 T (1 + \frac{1 - Q}{Q})$$

$$M(A) = \omega_1 \frac{T}{Q}$$

Since

$$\omega_1 = \frac{\tau_1}{\tau_1 + \tau_2} = \frac{Q}{T + Q}$$

we get that

$$M(A) = \frac{T}{Q} \frac{Q}{Q+T} = \frac{T}{Q+T}$$

Thus

$$M(A) = \frac{\prod_{j=1}^{n} a_j}{\prod_{j=1}^{n} a_j + \prod_{j=1}^{n} \bar{a}_j}.$$

We shall denote this operator as the **triple** $\prod$ aggregation operator.

We now show that this triple $\prod$ operator is a FIMICA operator. First, we see that it is commutative since the aggregation is independent of the indexing of the a_i. Secondly, we show that $g = 0.5$ is an identity element for this operator. Let

$$B = A \oplus \langle 0.5 \rangle$$

then

$$M(B) = \frac{g \prod_{j=1}^{n} a_j}{g \prod_{j=1}^{n} a_j + \bar{g} \prod_{j=1}^{n} \bar{a}_j}$$

Since for $g = 0.5$ we get that $\bar{g} = 0.5 = g$ we get

$$M(B) = M(A).$$

Finally we must prove monotonicity of the aggregation with respect to the a_i. Without loss of generality we shall show it is monotonic for a_1.

$$\frac{\partial M(A)}{\partial a_1} = \frac{(\prod_{j=1}^{n} a_j + \prod_{j=1}^{n} \bar{a}_j) \prod_{j=2}^{n} a_j - \prod_{j=1}^{n} a_j (\prod_{j=2}^{n} a_j - \prod_{j=2}^{n} \bar{a}_j)}{(\prod_{j=1}^{n} a_j + \prod_{j=1}^{n} \bar{a}_j)^2}$$

$$\frac{\partial M(A)}{\partial a_1} = \frac{(\prod_{j=1}^{n} a_j)(\prod_{j=2}^{n} a_j) + (\prod_{j=1}^{n} \bar{a}_j)(\prod_{j=2}^{n} a_j) - (\prod_{j=1}^{n} a_j)(\prod_{j=2}^{n} a_j) + (\prod_{j=2}^{n} \bar{a}_j)(\prod_{j=1}^{n} a_j)}{(\prod_{j=1}^{n} a_j + \prod_{j=1}^{n} \bar{a}_j)^2}$$

$$\frac{\partial M(A)}{\partial a_1} = \frac{(\prod_{j=1}^{n} \bar{a}_j)(\prod_{j=2}^{n} a_j) + (\prod_{j=2}^{n} \bar{a}_j)(\prod_{j=1}^{n} a_j)}{(\prod_{j=1}^{n} a_j + \prod_{j=1}^{n} \bar{a}_j)^2} \geq 0.$$

Thus we see that the above triple Π operator is indeed a FIMICA operator. Furthermore since its identity is 0.5 it is a full reinforcement operator.

Example: Aggregate A = <0.6, 0.2, 0.9, 0.8>. In this case

$a_1 = 0.6$ $a_2 = 0.2$ $a_3 = 0.9$ $a_4 = 0.8$

$\bar{a}_1 = 0.4$ $\bar{a}_2 = 0.8$ $\bar{a}_3 = 0.1$ $\bar{a}_4 = 0.2$

$$\prod_{j=1}^{4} a_j = 0.0864$$

$$\prod_{j=1}^{4} \bar{a}_j = 0.0064$$

$$M(A) = \frac{0.0864}{0.0864 + 0.0064} = 0.931.$$

Let us now investigate another form of aggregation model derivable from the preceding fuzzy system model. In implementing the fuzzy systems model describing our desired aggregation function we used the COA, center of area, defuzzification technique. An alternative defuzzification technique is the mean of maximal, MOM, method [Lee, 1990]. If we use the MOM method then we get

$$M(A) = T(a_1,....a_n) \quad \text{if } \tau_1 > \tau_2$$

$$M(A) = S(a_1,....a_n) \quad \text{if } \tau_1 < \tau_2$$

$$M(A) = \frac{1}{2}(T(a_1,....a_n) + S(a_1,....a_n)) \quad \tau_1 = \tau_2$$

We shall still retain our definition of *high* and *low* used in the antecedents,

$$L(x) = 1 - x$$

$$H(x) = x.$$

However now we shall use the Min operator instead of the product to aggregate the individual antecedent satisfactions thus

$$\tau_1 = \text{Min}_i[a\text{—}_i]$$

$$\tau_2 = \text{Min}_i[a_i]$$

Finally we shall assume that the consequent aggregation operators, T and S, are the Min and Max respectively. In this case we get that

$$M(A) = \text{Min}_i[a_i] \qquad \text{if } \text{Min}_i[a\text{—}_i] > \text{Min}_i[a_i]$$

$$M(A) = \text{Max}_i[a_i] \qquad \text{if } \text{Min}_i[a\text{—}_i] < \text{Min}_i[a_i]$$

$$\frac{1}{2}(\text{Min}_i[a_i] + \text{Max}_i[a_i]) \qquad \text{if } \text{Min}_i[a\text{—}_i] = \text{Min}_i[a_i]$$

It can be easily seen that this operator is commutative and monotonic with respect to the arguments.

We now show that this formulation has a fixed identity at g = 0.5. First we note that

$$\text{Min}_i[a\text{—}_i] = 1 - \text{Max}_i[a_i]$$

from De Morgan's Law. We now consider three situations

(1) **All $a_i > 0.5$.** In this case

$$\text{Min}_i[a_i] > 0.5$$

and

$$\text{Min}_i[a\text{—}_i] < 0.5.$$

Hence

$$M(A) = \text{Max}_i[a_i].$$

If we add an element equal to 0.5 then

$$\text{Min}_i[a_i] \wedge 0.5 = 0.5$$

$$\text{Min}_i[a\text{—}_i] \wedge 0.5 = 1 - (\text{Max}_i[a_i] \vee 0.5) < 1/2$$

and hence

$$M(A) = \text{Max}_i[a_i] \vee 0.5 = \text{Max}_i[a_i].$$

Thus no change has occurred with the addition of this element to our argument bag.

(2) **All $a_i < 0.5$.** In this case

$$\text{Min}_i[a_i] < 0.5$$

and

$$\text{Min}_i[a\text{—}_i] > 0.5.$$

Hence

$$M(A) = \text{Min}_i[a_i].$$

If we add an element equal to 0.5 then

$$\text{Min}_i[a_i] \wedge 0.5 < 0.5$$

$$\text{Min}_i[a\text{—}_i] \wedge 0.5 = 1 - (\text{Max}_i[a_i] \vee 0.5) = 0.5$$

and hence

$$M(A) = \text{Min}_i[a_i] \wedge 0.5 = \text{Min}_i[a_i]$$

and hence again no change has occurred as a result of the addition of this element.

(3) **Assume that $\text{Min}_i[a_i] < 0.5$ and $\text{Max}_i[a_i] > 0.5$.** In this case there exists at least one element in the argument greater then and at least one element less then 0.5. Furthermore we note that in this case $\text{Min}_i[a_i] \wedge 0.5 = \text{Min}_i[a_i]$ and $\text{Max}_i[a_i] \vee 0.5 = \text{Max}_i[a_i]$ and hence the evaluation doesn't change. All we need show is that the relationship between $\text{Min}_i[a_i]$ and $\text{Min}_i[a\text{—}_i]$ remains the same. In this case

$$\tau'_1 = \text{Min}_i[a\text{—}_i] \wedge 0.5 = 1 - (\text{Max}_i[a_i] \vee 0.5) = 1 - \text{Max}_i[a_i] = \tau_1$$

$$\tau'_2 = \text{Min}_i[a_i] \wedge 0.5 = \text{Min}_i[a_i] = \tau_2.$$

Thus again we see that no change occurs in the aggregation as a result of the addition of an argument equal to 0.5.

As a result of the above we see that the value 0.5 is a fixed identity for this operator and thus the above operator is a FIMICA operator with identity g = 0.5.

Example: Aggregate <0.4, 0.3, 0.6, 0.9>.

$a_1 = .4$ $\quad a\text{—}_1 = 0.6$

$a_2 = .3$ $\quad a\text{—}_2 = 0.7$

$a_3 = .6$ $\quad a\text{—}_3 = 0.4$

$a_4 = .9$ $\quad a\text{—}_4 = 0.1$

In this example

$$\text{Min}_i[a_i] = 0.3$$

$$\text{Min}_i[\bar{a}_i] = 0.1$$

Since $0.3 > 0.1$ then

$$M(A) = \text{Max}_i[a_i] = 0.9.$$

4.Noncommutative self identity operators

4.1 Aggregation operators with self identity property.

In many domains we are faced with the problem of aggregating a collection of numerical readings to obtain a so called mean or typical value. Formally. if $x_1,\ldots., x_n$ is a collection of readings then we shall denote this process as

$$Agg(x_1,\ldots., x_n) = a_n$$

In deciding upon the form of the aggregation operator a number properties should be associated with this operation. These properties are generally based upon natural considerations associated with the idea of a typical value. A first property is that which we call **natural boundary** [Yager, 1993].

Definition 4.1: An aggregation operator is said to have a natural boundary if

$$Agg(a) = a$$

This effectivity says that if we just have one reading that should be our typical value. A second property associated with this kind of aggregation operation is that of **self identity** [Yager, 1994].

Definition 4.2: An aggregation operator is said to be a self identity operator if it posses the following property. Assume

$$a = Agg(x_1,\ldots., x_n)$$

then

$$Agg(x_1,\ldots., x_n, a) = Agg(x_1,\ldots., x_n) = a$$

Thus we see that in the case of self identity operators adding an element equal to the already established value doesn't change the aggregation value.

Another reasonable property that one can associate with the processors of finding or typical value is that of **monotonicity**.

Definition 4.3: An aggregation operator is said to the monotone if

$$Agg(x_1, x_2\ldots., x_n) \geq Agg(y_1, y_2, \ , y_n) \text{ when } x_i \geq y_i \text{ for all i.}$$

The imposition of these three conditions bring along some other salient properties. First we recall that an aggregation operator is called idempotent if

$$Agg(x_1,\ldots,x_n) = a \qquad \text{if } x_i = a \text{ for all } i.$$

That is if all the elements in the aggregation are the same then this should be our typical value. The following result shows that our first three requirement assure us of this condition.

Theorem 4.1: Assume Agg is in operator that has a natural boundary and is a self identity operator then Agg is idempotent.

Proof: From natural boundary Agg(a) = a. Consider Agg(a, a), from the self identity property this must also evaluate to a. Continuing in this manner we can easily see that idempotency is guaranteed.

We should note that idempotency doesn't guarantee self identity. Thus self identity and natural boundary are stronger then idempotency.

The assumption of idempotency, self identity and natural boundary, and monotonicity guarantees that the typical value always lies between the maximum and minimum of the values aggregated. Consider $Agg(x_1,\ldots,x_n)$ and let

$$a = Max_i[x_i]$$

and let

$$b = Mini[x_i]$$

then from monotonicity

$$Agg(a, a,\ldots, a) \le Agg(x_1, \ldots, x_n) \le Agg(b,.b, \ldots b)$$

and from idempotency

$$a \le Agg(x_1, \ldots, x_n) \le b$$

Another salient property guaranteed by self identity and monotonicity is expressed in the following [Yager, 1993].

Theorem 4.2: Assume Agg is a monotone and self identity operator then

$$Agg(x_1, \ldots, x_n, K) \ge Agg(x_1, \ldots, x_n) \qquad \text{if } K > Agg(x_1,..x_n)$$

$$Agg(x_1, \ldots, x_n, K) \le Agg(x_1, \ldots, x_n) \qquad \text{if } K < Agg(x_1,...x_n)$$

Proof: Assume $a = Agg(x_1, \ldots, x_n)$ and consider $Agg(x_1, \ldots, x_n, K)$. We first note that form

self identity if K = a then $Agg(x_1, \ldots, x_n, K) = a$. From monotonicity if K > a then $Agg(x_1, \ldots, x_n, K) \geq a$ and if K < a then from monotone $Agg(x_1, \ldots, x_n, K) \leq a$.

Essentially this theorem implies that values greater then the already established value tend to increase the score while those below it tend to decrease this score.

Another condition often associated with aggregation operators is that of commutativity or symmetry. This condition essentially implies that the indexing (ordering) of the arguments doesn't matter. However, in many situations where we provide aggregation this doesn't necessarily hold. Consider a situation in which our arguments are temporal in nature, in this case x_i indicates the i^{th} observed reading. In situations in which we feel that the basic underlying process generating the readings is changing we may desire to give more emphasize to the later readings rather then to the former ones. Another case in which a distinction is made between the indexing of elements occurs in the domain a human interaction. An often observed phenomena in human interaction is that of *first impression*. As a result of this phenomena earliest observations create a kind of anchor (impression) from which it is difficult for subsequent observations to change. In this case it is the earlier observation that is counted more. For example, let each i indicate and encounter with another person. For a given encounter let $x_i \in [0, 1]$ indicate the degree to which we observe the other person to be *generous*, for example. If $Agg(x_1, \ldots x_n)$ indicates our overall evaluation of this persons generosity then if the "first impression" phenomena is in effect it is our earliest observations on the persons generosity that count the most.

In defining our aggregation operator we required the following three properties:

(1) **natural boundaries**

(2) **self identity**

(3) **monotonicity**

We also should that properties (1) and (2) imply that the operator must be idempotent. We also showed that idempotency and monotonicity require that the aggregation is bounded by the max and min of the arguments.

Another property not required was that of associativity. We recall associativity is

$$Agg(x, y, z) = Agg(x, Agg(y, z)) = Agg(Agg(x, y), z).$$

A significant benefit of having associativity is that it provides a consistent way of extending the aggregation process starting with the aggregation of just two elements. Implicit in the assumption of associativity is a consistent way of going from the aggregation of n elements to n + 1 elements. That is if Agg. is associative

$$Agg(a_1, \ldots, a_n, a_{n+1}) = Agg(Agg(a_1, \ldots, a_n), a_{n+1}).$$

Thus associativity mandates how we extend our operator and does so in a way that keeps some consistency between the aggregation of n and n + 1 elements. In particular monotonicity and idempotency along with associativity always assures us that $Agg(a_1, \ldots, a_n, a_{n+1})$ always is bounded by $Agg(a_1, \ldots, a_n)$ and a_{n+1}.

So we see that associativity is a desirable property. However as the following theorem shows monotonicity and idempotency in general precludes associativity.

Theorem 4.3: If Agg is an idempotent and monotonic operator, not equal to the max or min operators, then Agg is not associative.

Proof: Assume x > y. Then

$$x \leq Agg(x,y) \leq y$$

and since Agg is not the max or min operator we have

$$x < Agg(x,y) \prec y.$$

From monotonicity we have

$$Agg(Agg(x, y), y) > Agg(x, y) > x.$$

If we assume associativity

$$Agg(Agg(x,y), y) = Agg(x, Agg(y, y)),$$

however from idempotency we get

$$Agg(Agg(x, y), y) = Agg(x, y)$$

which is a contradiction.

Thus by requiring our aggregation operator to satisfy monotonicity and idempotency we can't in general impose associativity, unless we only use the max or min operator. With just idempotency and monotonicity and no associativity no natural way exists of extending our operator

is imposed upon us. Thus for example we can satisfy our conditions of monotonicity and idempotency and have radically different operations for each degree of cardinality of aggregation. That is we could have an aggregation operator such that

$$Agg(a_1, a_2) = Min[a_1, a_2]$$

$$Agg(a_1, a_2, a_3) = \frac{1}{3} \sum_{i=1}^{R} a_i$$

$$Agg(a_1, a_2, a_3, a_4) = Max_i [a_i]$$

$$Agg(a_1, a_2, a_3, a_4, a_5) = Median_i [a_i]$$

Thus in situations in which we require the aggregation operator to be idempotent and monotonic we have no restriction on the process of extending the operator to greater cardinalities in a consistent way.

It should be noted that the requirement of self identity, which along with natural boundaries implies the idempotency, brings along with it some requirements for consistency in extending an operators cardinality. For we note the self identity property relates the aggregation process at cardinalities n and n + 1 in such a way that adding as the n + 1th element the aggregated value of the previous n elements must give us the original aggregated value. Thus it is anticipated, that by requiring the stronger conditions of self identity and natural boundaries rather then simply idempotency will help in obtaining a consistent extension. In particular we anticipate that the requirement of these two conditions will retrieve some of burden of not having associativity.

A further relief from the lack of associativity can be had if we impose some a priori specification on the form of aggregation that is to be used in the general case.

4.2 Linear Self Identity Aggregation Operators

As we note without the associativity property in order try to attain the goal of consistency in the aggregation process we should a priori indicate some general uniform formulation for the aggregation for each cardinality of the argument. In the following we shall consider the class of aggregation operators of the form

$$Agg(x_1, \ldots, x_n) = \sum_{j=1}^{n} w_{nj}\ x_j$$

In the above w_{nj} is the weight associated with the j^{th} argument in the case when there are n arguments to be aggregated. In selecting the weights for each case we must do so in a way that we satisfy the three conditions required; natural boundaries, self identity and monotonicity. The inherent idempotency implied by these conditions requires that for any n

$$Agg(x_1, \ldots, x_n) = \sum_{j=1}^{n} w_{nj}\ x_j = a$$

if all $x_j = a$. Thus this condition implies that

$$a \sum_{j=1}^{n} w_{nj} = a$$

and hence for any n

$$\sum_{j=1}^{n} w_{nj} = 1$$

Secondly the monotonicity condition requires that all the $w_{nj} \geq 0$ for all n and j. We see this from the following. Without loss of generality assume $w_{n1} < 0$. Consider the arguments $a_i, \ldots, a_n$ and $b_i, \ldots, b_n$, where

$$a_i = b_i = 0 \qquad i = 2,.....n$$

and

$$a_1 > b_1 > 0.$$

In this case

$$\Sigma\, w_{nj}\, b_j > \Sigma\, w_{nj}\, a_j$$

and monotonicity is not satisfied

Thus we see that the weights must satisfy

$$(1) \sum_{j=1}^{n} w_{nj} = 1 \qquad \text{for all n}$$

$$(2)\ w_{nj} \geq 0 \qquad \text{for all n and j}$$

We see that we still have considerable freedom in selecting the weights.

If we impose the condition of commutativity we uniquely specify the weights for each

aggregation cardinality. The property of commutativity requires that for each cardinality n

$$w_{nj} = w_{ni} = K_n \qquad \text{for all i and j.}$$

Furthermore from condition one above we get

$$\sum_{j=1}^{n} K_n = 1$$

and hence

$$K_n = \frac{1}{n},$$

and thus the weights are uniquely specified.

However as we indicated earlier we are interested in aggregation situations in which the assumption of commutativity is not imposed. As we shall see in the following the requirement of self identity provides some relief by imposing a very strong and useful constraint on the selection of the weights while still giving us enough freedom to easily specify some information on how we should relate the indexed arguments in lieu of commutativity.

We now look at the requirements imposed upon the weights in the aggregation stemming from the necessity to satisfy the condition of self identity.

In the situation in which n = 1, from the fact that the weights must sum to one, we see that $w_{11} = 1$. Consider now the case in which n = 2, here we have

$$Agg(x_1, x_2) = w_{21}\, x_1 + w_{22}\, x_2$$

The condition of self identity requires that

$$w_{21}\, x_1 + w_{22}\, (w_{11}\, x_1) = w_{11}\, x_1$$

for all x_1. Hence we have

$$w_{21} + w_{22}\, w_{11} = w_{11}$$

$$w_{22} = \frac{w_{11}}{w_{11}} - \frac{w_{21}}{w_{11}} = 1 - \frac{w_{21}}{w_{11}}.$$

In addition we have

$$w_{21} + w_{22} = 1$$

This specifying the ratio of $\frac{w_{21}}{w_{11}}$ uniquely determines w_{21} and w_{22}.

Consider now the case in which n = 3, here we have

$$Agg(x_1, x_2, x_3) = w_{31}\, x_1 + w_{32}\, x_2 + w_{33}\, x_3$$

From the self identity condition we have

$$w_{31}\, x_1 + w_{32}\, x_2 + w_{33}(w_{21}\, x_1 + w_{22}\, x_2) = w_{21}\, x_1 + w_{22}\, x_2$$

Since this must hold for all x_1 and x_2 we get

$$w_{31} + w_{33}\, w_{21} = w_{21}$$

$$w_{32} + w_{33}\, w_{22} = w_{22}$$

From this we see that

$$w_{33} = \frac{w_{21} - w_{31}}{w_{21}} = 1 - \frac{w_{31}}{w_{21}}$$

$$w_{32} = w_{22}(1 - w_{33}) = w_{22}\,\frac{w_{31}}{w_{21}}$$

Furthermore since $w_{21} + w_{22} = 1$ we have

$$w_{32} = (1 - w_{21})\,\frac{w_{31}}{w_{21}} = \frac{w_{31}}{w_{21}} - w_{31}$$

In addition the condition

$$w_{31} + w_{32} + w_{33} = 1$$

assures us that the weights are uniquely determined by selecting w_{31}. It is interesting to observe since $w_{11} = 1$ then

$$w_{33} = 1 - \frac{w_{31}}{w_{21}}$$

$$w_{32} = \frac{w_{31}}{w_{21}} - \frac{w_{31}}{w_{11}}$$

In particular the weights are uniquely determined by the ratio of w_{31} with first weights in the previous aggregations.

For the case when we have n arguments to

$$Agg(x_1,\ldots x_n) = w_{n1}\, x_1 + w_{n2}\, x_2 + \ldots\ldots w_{nn}\, x_n$$

From the self identity property we have

$w_{n1}\, x_1 + w_{n2}\, x_2 + \ldots w_{nn}(w_{n-1,1}\, x_1 + w_{n-1,2}\, x_2 + \ldots w_{n-1,n-1}\, x_{n-1}) = w_{n-1,1}\, x_1 + w_{n-1,2}\, x_2 + \ldots + w_{n-1,n-1}\, x_{n-1}$

Since this must hold for all values of $x_1, \ldots, x_{n-1}$ we have

$$w_{n1} + w_{nn}\, w_{n-1,1} = w_{n-1,1} \qquad (1)$$

$$w_{n2} + w_{nn}\, w_{n-1,2} = w_{n-1,2} \qquad (2)$$

$$w_{n3} + w_{nn}\, w_{n-1,3} = w_{n-1,3} \qquad (3)$$

.....

$$w_{nj} + w_{nn}\, w_{n-1,j} = w_{n-1,j} \qquad (j)$$

.....

$$w_{nn-1} + w_{nn}\, w_{n-1,n-1} = w_{n-1,\,n-1} \qquad (n-1)$$

Solving these equations we get

$$w_{n2} = w_{n1}\left(\frac{1}{w_{21}} - \frac{1}{w_{11}}\right)$$

$$w_{n3} = w_{n1}\left(\frac{1}{w_{31}} - \frac{1}{w_{21}}\right)$$

$$w_{n4} = w_{n1}\left(\frac{1}{w_{41}} - \frac{1}{w_{31}}\right)$$

$$w_{ni} = w_{n1}\left(\frac{1}{w_{i_1}} - \frac{1}{w_{i-1,1}}\right)$$

$$w_{nn} = 1 - \frac{w_{n1}}{w_{n-1,\,1}}$$

Furthermore we have the condition that

$$\sum_{i=1}^{n} w_{n,i} = 1$$

From this we see that

$$w_{n,1} + \sum_{i=2}^{n} \frac{w_{n1}}{w_{i_n}} - \sum_{i=1}^{n-1} \frac{w_{n1}}{w_{i1}} = 1$$

$$w_{n,1} + 1 - \frac{w_{n_1}}{w_{11}} = 1$$

and hence

$$w_{n,1} = \frac{w_{n1}}{w_{11}}$$

The key observation to be made here is that the imposition of the requirement of self identity puts a strong and useful restriction on the formulation of the weights. In particular we see that for $i > 1$ all the w_{ni} are determined from the simple ratio of the first element in the n^{th} order aggregation w_{n1}, with the first elements of the preceding order aggregation. In particular we can uniquely determine our weights by simply specifying a formulation for these ratios. Furthermore, we also note that the condition that the weights be positive implies that

$$\frac{1}{w_{i,1}} - \frac{1}{w_{i-1,1}} \geq 0$$

and thus

$$w_{i-1,1} \geq w_{i,1}.$$

That is we must have

$$\frac{w_{i-1,1}}{w_{i,1}} \leq 1$$

4.3. On Some Classes of Linear Self Identity Operators

In the previous section we showed that for the case of linear aggregation having self identity in addition to natural boundaries and monotonicity the weights associated with the aggregation are uniquely determined by simply specifying the ratio of the first weights in the aggregation.

The formulation chosen for ratio is a reflection of the type of non- commutativity we desire to capture by our aggregation model. In this section we shall consider some special classes of aggregation operators.

A first class of these operators we could consider are those that are commutative, the effect of all the arguments at each cardinality must be the same. Actually the requirements idempotency along with commutativity uniquely specify the weights. For any n commutativity requires

$$w_{nj} = K_n \qquad \text{for all } j$$

since

$$\Sigma\, w_{n,j} = 1$$

we must have

$$\sum_{i=1}^{n} K_n = 1$$

here

$$K_n = 1/n$$

Thus in the case of a commutative operator we get that uniquely the weights are 1/n. In the following we shall look at formulas of non- commutative operators.

We now consider the case in which we impose the following additional restriction of the weights, for all i

$$\frac{w_{i,1}}{w_{i-1,1}} = \alpha$$

where $\alpha \in [0,1]$. That is, we require that the first weight in each succeeding aggregation is reduced

by α portion. In this case we have that

$$\frac{w_{n,1}}{w_{n-1,1}} = \alpha.$$

Since

$$\frac{w_{n,1}}{w_{n-2,1}} = \frac{w_{n,1}}{w_{n-1,1}} \frac{w_{n-1,1}}{w_{n-2,1}}$$

we get

$$\frac{w_{n,1}}{w_{n-2,1}} = \alpha^2$$

More generally we have

$$\frac{w_{n,1}}{w_{n-j,1}} = \alpha^j$$
$$\frac{w_{n,1}}{w_{i,1}} = \alpha^{n-i}$$

From this we see that the weights for i = 2, .,.,., n can be easily obtained as

$$w_{n,i} = (\frac{w_{n,1}}{w_{i,1}} - \frac{w_{n,1}}{w_{i-1,1}}) = \alpha^{n-i} - \alpha^{n-i+1} = \alpha^{n-i}(1-\alpha)$$

and

$$w_{n,1} = \frac{w_{n,1}}{w_{11}} = \alpha^{n-1}$$

Using the weight obtained we get as our aggregation function

$$a_n = \alpha^{n-1} x_1 + \sum_{i=2}^{n} \alpha^{n-i}(1-\alpha) x_i$$

As we show in the following a recursive formulation can be obtained. Since

$$a_{n+1} = \alpha^n x_1 + \sum_{i=2}^{n+1} \alpha^{(n-i+1)}(1-\alpha) x_i$$

and therefore

$$a_{n+1} = \alpha(\alpha^{n-1} x_1 + \sum_{i=2}^{n} \alpha^{n-i}(1-\alpha) x_i) + (1-\alpha)x_{n+1}$$

we have after substituting the form for a_n

$$a_{n+1} = \alpha\, a_n + (1-\alpha)\, x_{n+1}$$

Thus a_{n+1} can be simply obtained for knowledge of a_n, x_{n+1} and α. We note this form of aggregation known as exponential smoothing [Brown, 1963] is a classic noncommutative type of linear aggregation.

Two special cases are worth pointing out. If $\alpha = 1$ then

$$w_{n,1} = 1$$

$$w_{n,i} = 0 \qquad \text{for } i = 2,.....n.$$

Thus in this case we always obtain as our aggregated value the first observation. This case can be seen to correspond to a very strong very impression phenomena.

If $\alpha = 0$ then

$$w_{n,n} = 1$$

$$w_{n,i} = 0 \qquad \text{for all } i = 1,...n - 1$$

In this case our aggregated value is always our latest observation.

We now consider a different relationship between the weights, one in which for all i

$$\frac{w_{i,1}}{w_{i-1,1}} = \left(\frac{i-1}{i}\right)^K$$

where $K \geq 0$. In this case it is easy to see that

$$\frac{w_{n,1}}{w_{i,1}} = \left(\frac{n-1}{n}\right)^K \left(\frac{n-2}{n-1}\right)^K \cdots \left(\frac{i}{i+1}\right)^K = \left(\frac{i}{n}\right)^K$$

From this we see that for any cardinality of aggregation the associated weights for i = 2, . . , n are

$$w_{n,i} = \left(\frac{w_{n,1}}{w_{i,1}} - \frac{w_{n,1}}{w_{i-1,1}}\right) = \frac{i^K}{n^K} - \frac{(i-1)^k}{n^K} = \frac{i^K - (i-1)^K}{n^K}$$

Furthermore $w_{n,1}$ is

$$w_{n,1} = \frac{w_{n,1}}{w_{11}} = \frac{1}{n^K}$$

We note that we can express this as

$$w_{n,1} = \frac{i^K}{n^K} - \frac{(i-1)^K}{n^K}$$

where i = 1. Since this the same structure as the above we have for all i = 1,...., n

$$w_{n,i} = \frac{i^K - (i-1)^K}{n^K}$$

Let us look at the form of the weights as K moves form zero to infinity. For K = 0, we get

$$w_{n,1} = 1$$

$$w_{n,i} = 0 \qquad i \neq 1$$

This we have a situation in which we only consider the first element in the aggregation, i.e.

$$a_n = x_1 \qquad \text{for all n.}$$

This is an extreme example of a backward looking type aggregation, strong first impression.

For the case when $K \rightarrow \infty$ we get

$$w_{n,n} = 1$$

$$w_{n,i} = 0 \qquad \text{for all other i}$$

Thus in this case we get

$$a_n = x_n$$

This an extreme example of a forward looking linear aggregation.

Let us now consider the case in which K = 1. Here we get

$$w_{n,i} = \frac{i - (i - 1)}{n} = \frac{1}{n} \qquad \text{for all i.}$$

This gives us the simple averaging operating which is neutral regarding backward/forward looking. It is interesting to note that this simple case occur when

$$\frac{w_{i,1}}{w_{i-1,1}} = \frac{i - 1}{i}$$

One other special case is worth noting, K = 2. In this case

$$w_{n,i} = \frac{i^2 - (i - 1)^2}{n^2} = \frac{2i - 1}{n^2}$$

We particularly note that in this case the weights are linear with respect to i,

$$a_n = \sum_{i=1}^{n} \left(\frac{2i - 1}{n^2}\right) x_i = \frac{1}{n^2}\left(\sum_{i=1}^{n} 2i\, x_i - \sum_{i=1}^{n} x_i\right)$$

Here we see that for K = 2 we emphasize the later elements in the aggregation, hence this can be considered a forward looked type aggregation.

More generally we see that the value of K adjudicates between the emphasis on the earlier arguments versus the later elements, determines the type of noncommutativity. As K moves form zero to infinity we move form emphasizing the earlier elements to emphasizing the later elements. If particular for k < 1 it is backward looking and for K > 1 it is forward looking. When K = 1 we get the ordinary average which is commutative.

A general recursive formulation for this aggregation can be obtained which greatly simplifies the calculations. We first note that in this case

$$a_n = \sum_{i=1}^{n} w_{n,i} X_n = \sum_{i=1}^{n} \left(\frac{i^K}{n^K} - \frac{(i - 1)^K}{n^K}\right) x_i$$

also we note that

$$a_{n+1} = \sum_{i=1}^{n+1} \frac{i^K}{(n+1)^K} x_i$$

From this it follows that

$$a_{n+1} = \left(\frac{n+1}{n+1}\right)^K - \left(\frac{n}{n+1}\right)^K) x_{n+1} + \frac{1}{(n+1)^K} \sum_{i=1}^{n} ((i)^K - (i-1)^K) x_i$$

$$a_{n+1} = (1 - \left(\frac{n}{n+1}\right)^K) x_{n+1} + \frac{n^k}{n^k(n+1)^K} \sum_{i=1}^{n} ((i)^K - (i-1)^K) x_i$$

$$a_{n+1} = (1 - \left(\frac{n}{n+1}\right)^K) x_{n+1} + \frac{n^k}{(n+1)^K} a_n$$

We now consider the case in which the noncommutativity is expressed by

$$\frac{w_{i,1}}{w_{i-1,1}} = \frac{\log_2(i)}{\log_2(i+1)}$$

For this case we get

$$\frac{w_{n,1}}{w_{i,1}} = \frac{\log_2(i+1)}{\log_2(n+1)}$$

Using our developed relationship for the weights associated with this aggregation we for i = 2,..n

$$w_{n,i} = \left(\frac{\log_2(i+1)}{\log_2(n+1)} - \frac{\log_2(i)}{\log_2(n+1)}\right)$$

and

$$w_{n,1} = \frac{1}{\log_2(n+1)}$$

however

$$\frac{1}{\log_2(n+1)} = \left[\frac{\log_2(i+1)}{\log_2(n+1)} - \frac{\log_2(i)}{\log_2(n+1)}\right] \qquad \text{for } i = 1$$

Thus we see that for all i = 1,...., n

$$w_{n,i} = \left|\frac{\log_2(i+1)}{\log_2(n+1)} - \frac{\log_2(i)}{\log_2(n+1)}\right|$$

Furthermore we see that we can express $w_{n,i}$ as

$$w_{n,i} = \frac{1}{\log_2(n+1)} \left(\log_2\left(\frac{i+1}{i}\right)\right) = \frac{1}{\log_2(n+1)} \left(\log_2\left(1 + \frac{1}{i}\right)\right).$$

Using the Mac Claurin series expansion [Hart, 1957] we get

$$\log_2\left(1 + \frac{1}{i}\right) \approx \frac{1}{i} \log_2 e$$

and hence for all i

$$w_{n,i} \approx \frac{1}{i} \frac{\log_2 e}{\log_2(n+1)}$$

Thus we see that the weights are approximately proportional to the inverse of the index.

Furthermore we note that for the aggregated value

$$a_n \approx \frac{\log_2 e}{\log_2(n+1)} \sum_{i=1}^{n} \frac{x_i}{i}$$

Thus we see that this type of noncommutativity emphasizes the early elements over the later ones and leads to a kind of backward looking aggregation.

In previous cases we assumed that the weights where independent of the arguments here we shall relax that assumption and allow the weights to be dependent upon the arguments. However we shall restrict ourselves to situations in which the arguments lie in the unit interval. In the following we shall assume that the specification for the type of noncommutativity is expressed by

$$\frac{w_{n,1}}{w_{i,1}} = \frac{S(x_1,\ldots, x_i)}{S(x_1,\ldots, x_n)}$$

where S is any t- conorm operator [Dubois, Prade, 1985]. For simplicity we shall denote $S(x_1,\ldots, x_i)$ as S_i. Using our formulation for weights we get

$$w_{n,j} = \frac{1}{S_n}(S_j - S_{j-1}) \qquad \text{for } j = 2,\ldots..,\ n$$

$$w_{n,1} = \frac{S_1}{S_n}$$

If we denote $S_0 = S(\,) = 0$ then we can express

$$w_{n,1} = \frac{1}{S_n}(S_1 - S_0)$$

and thus for all $i = 1,\ldots..,\ n$

$$w_{n,j} = \frac{1}{S_n}(S_j - S_{j-1}).$$

In this case the aggregated value is

$$a_n = \sum_{j=1}^{n} w_{n,\,j}x_j = \frac{1}{S_n}\sum_{j=1}^{n}(S_j - S_{j-1})x_j$$

and

$$a_{n+1} = \frac{1}{S_{n+1}}\sum_{j=1}^{n+1}(S_j - S_{j-1})x_j$$

Furthermore we can express a_{n+1} as

$$a_{n+1} = \frac{S_n}{S_{n+1}}\frac{1}{S_n}\sum_{j=1}^{n}(S_j - S_{j-1})x_j + \frac{1}{S_{n+1}}(S_{n+1} - S_n)\,x_{n+1}$$

which leads to the recursive form

$$a_{n+1} = \frac{S_n}{S_{n+1}}a_n + (1 - \frac{S_n}{S_{n+1}})\,x_{n+1}$$

We note that the aggregation process stops and attains it final value as soon as we have $S_j = 1$ for some j. If $S_j = 1$ then from the properties of the t-conorm we have $S_{j+1} = 1$ and hence $\frac{S_j}{S_{j+1}} = 1$ and therefore

$$a_{j+1} = a_j + (1-1)\, x_{j+1} = a_j.$$

It is interesting to consider the special case when S = Max. In this situation since

$$w_{n,\,j} = \frac{1}{S_n}\,(S_j - S_{j-1})$$

we get

$$w_{n,j} = 0 \qquad \text{if } x_j \le \max x_i \text{ for } i = 1,\ldots,\, j-1$$

Thus we only consider get contributions from components that are strictly greater then all the previous components. Furthermore when $w_{n,j} \neq 0$ it is the case that $S_j = x_j$. Consider the collection $\{x_1,\ldots,\, x_n\}$. We shall say an element x_j in this collection is **locally maximal** if

$$x_j > \text{Max } x_i \quad \text{for } i = 1,\ldots,\, j-1.$$

If we let $y_1,\ldots,\, y_k$ be the collection at locally maximal elements where y_i is the i^{th} locally maximal element, then we see that

$$a_n = \frac{1}{y_K} \sum_{i=1}^{K} ((y_i - y_{i-1})\, y_i) = \sum_{i=1}^{K} (y_i^2 - y_{i-1}\, y_i)$$

where we assume $y_0 = 0$.

As we previously noted the aggregation process stops when $S_n = 1$, for S = Max this occurs when we have some argument with value 1.

It should be noted that the ordered collection of locally maximal elements form a kind of step function and that the aggregation is strongly dependent upon this step function. Consider a string of arguments $x_1,\ldots,\, x_n$ with local maximal's $y_1,\ldots,\, y_k$. We see that if $y_1 = 1$, then K = 1 and $a_j = 1$ for all j. If $y_1 \neq 1$ then it is always the case that $a_n < 1$. We see this as follows. First we recall that

$$a_n = \frac{1}{y_K} \sum_{i=1}^{K} ((y_i - y_{i-1})y_i)\,.$$

Assume $y_1 = b$, then

i. If K = 1, we have $a_n = b$.

ii. If K > 1, then let $y_k = d$ in this case

$$a_n = \frac{1}{y_k} \sum_{i=1}^{K} (y_i - y_{i-1})\, y_i \le \frac{b}{d}\,.\, b + \frac{1}{d} \sum_{i=2}^{K} (y_i - y_{i-1})\, d$$

Furthermore

$$\sum_{i=2}^{K} y_i - y_{i-1} = d - b$$

and hence

$$a_n \leq \frac{b^2}{d} + d - b$$

$$a_n \leq d - \frac{1}{d}(bd - b^2) < d$$

Another interesting case occurs when we use the Lukasiewiscz t - conorm [Dubois, Prade,1985]

$$S(x_1,..., x_n) = \text{Min}[1, \sum_{i=1}^{n} x_i]$$

In this case

$$w_{n,j} = \frac{1}{S_n}(S_j - S_{j-1})$$

and if $S_j < 1$ then

$$w_{n,j} = \frac{1}{S_n}(x_j).$$

If $S_{j-1} \geq 1$ then

$$w_{n,j} = 0$$

and $S_j = 1$ and S_j-1 < 1 then

$$w_{n,j} = 1 - S_{j-1}$$

Essentially a_n is equal to the square of the first K elements where K is the value where the sum of the arguments reaches one.

5. Understanding the Median as a Fusion operator.

5.1. Characteristics of Fusion Operators

Here we shall briefly describe some of the characteristics associated with fusion operators. We shall assume that V is some attribute whose value we are trying to ascertain. In addition we shall assume that the value of V lies in the space X. Furthermore, we shall assume that we have a collection of readings $D = \{a_1, a_2, ..., a_n\}$ which purport to be the value of V. In this environment the problem of information fusion becomes one of obtaining some function F which takes the observations in D and returns some value as the fused value, thus

$$F(a_1, a_2, ..., a_n) = x.$$

Our concern here is to try to investigate the formulation of the function F.

We shall first indicate a number of properties which are useful in characterizing any proposed fusion function. One property that one can associate with F is that of **idempotency**. This property requires that if all the observations in D are the same F should return this value

$$F(a, a, ..., a) = a.$$

In particular this characteristic can be seen as a required property of any fusion operator of the type we are investigating.

Another characteristic associated with fusion operators is that commutativity or symmetry. This property implies that all observations in D are treated in the same manner, essentially it says that the indexing of the observations is not important. A number of situations exists in which the imposition of this condition may not be warranted. One case is that in which the observations have different importances associated with them. A second situation occurs when there is some temporal aspect to the observations.

In constructing a fusion operator we of course desire that the fused value in some way reflect the values of the observation. That is, we would expect some positive association between the elements in the bag D and the fused value x. For example, if the observation collection $D = \{a_1, a_2, ..., a_n\}$ returns a fused value of x we would anticipate that if we add to this collection

an observation equal to x we should still get as our fused value x. This is called a **self identity property** [Yager, 1994]. In domains in which their exists some ordering among the elements in X a more formal manifestation of this positive association can be expressed via the concept of monotonicity. A fusion operator F is called **monotonic** if

$$F(a_1, a_2, ..., a_n) \geq F(b_1, b_2, ..., b_n)$$

if $a_i \geq b_i$ for all i = 1...n.

An operator F that has the properties of idempotency, commutativity and monotonicity is called a mean operator [Dubois, Prade, 1985]. This operator has been considerably studied in the literature and has a large number of members of its family. One important property exhibited by this class is that [5, 6]

$$\text{Min}_i [a_i] \leq F(a_1, a_2, ..., a_n) \leq \text{Max}_i [a_i].$$

5.2. Penalty Based Fusion

In any application in which we are faced with the problem of fusing observations we must provide some mechanism for selecting among the large class possible fusion operators the one that best satisfies the needs of the domain. In the following we shall suggest one such approach to accomplishing this task.

Consider the observation a_i and a fused value x. We can associate with each of these pairs a function $P(a_i, x)$ into the non-negative real numbers called the local penalty function. In essence this function assigns a penalty to the fusion process based upon the compatibility of the observation a_i and the fused value x. It says that if you conclude something that doesn't agree with an observation you are penalized by that observation. Furthermore, we desire that if $x = a_i$ then no local penalty is incurred. However, if $x \neq a_i$ then some penalty should be incurred. This penalty should in some way reflect how far away x is from a_i. In particular, the penalty should not decrease as the distance between a_i and x increases.

Formally we represent

$$P(a, x) \rightarrow \hat{r} \qquad (\hat{r} \text{ non-negative real numbers})$$

where

1. $P(a, x) = 0$ if $a = x$
2. $P(a, x) > 0$ if $a \neq x$
3. $P(a, x) \geq P(b, x)$ if $|a - x| > |b - x|$

Having introduced this local penalty function, a measure of penalty associated with each of the observations, we must now consider an overall penalty function. The overall penalty function Pen(D, x) is defined on the collection of observation $D = \{a_1, a_2, ..., a_n\}$. We let

$$Pen(D, x) = \sum_{i=1}^{n} P(a_i, x)$$

This function measures the total penalty incurred if we say that x is the fused value of the data collection D.

In this environment we can consider as the best (optimum) fused value of the elements in D the value x that minimizes this total penalty function. Thus the fused value of D, $F(a_1, a_2, ..., a_n)$ is obtained as the value x such that

$$Pen(D, x) = Min_{y \in X} Pen(D, y)$$

We now consider some of the properties associated with such a fusion process. It is easy to see that this must be a commutative aggregation. Next we show that the above process always leads to an idempotent aggregation. Assume $a_i = a$ for all i. In this case if we select x = a then for each i, $P(a_i, x) = 0$ and hence Pen(D, x) = 0. If $x \neq a$ then $P(a_i, x) > 0$ for all i and hence Pen(D, x) > 0. Thus the aggregation will select a and is idempotent.

We now show that this leads to a fusion function where

$$Min_i[a_i] \leq F(a_1, a_2, ..., a_n) \leq Max_i[a_i]$$

Consider some value $z < Min_i[a_i]$. In this case

$$Pen(D, z) = \sum_{i=1}^{n} P(a_i, z)$$

Without loss of generality let $a_1 = Min_i\, a_i$. Consider now the fused value a_1. In this case

$$Pen(D, a_1) = \sum_{i=1}^{n} Pen(a_i, a_1)$$

Since for all i = 2, ...n, $|a_1 - a_i| < |z - a_i|$ then $P(a_i, a_1) \leq P(a_1, z)$. Furthermore, $P(a_1, z) > 0$ and $P(a_1, a_1) = 0$ from this it follows that

$$Pen(D, a_1) < Pen(D, z)$$

and thus z can't be the fused value. In a similar manner we can show that a value greater then Max_i a_i can't be the fused value.

We now show that a fusion operator determined in the preceding manner always exhibits a self identity property. Let $D = \{a_1, a_2, ..., a_n\}$ and assume x^* is the fused value based upon the preceding procedure. Thus we know that

$$Pen(D, x^*) = Min_x \sum_{i=1}^{n} P(a_i, x)$$

Consider now the addition of the observation x^*. In this case our data set is $D' = \{a_1, a_2, ...a_n, x^*\}$. To find the fused value we must find the value x that minimizes

$$Pen(D', x) = \sum_{i=1}^{n} P(a_i, x) + P(x^*, x).$$

If we select $x = x^*$ then

$$Pen(D', x^*) = Pen(D, x^*) = Min_x \left(\sum_{i=1}^{n} P(a_i, x) \right)$$

If $x = z \neq x^*$ then

$$Pen(D', z) = \sum_{i=1}^{n} P(a_i, z) + P(x^*, z)$$

Since $P(x^*, z) > 0$ and $\sum_{i=1}^{n} P(a_i, z) > Pen(D, x^*)$

we have

$$Pen(D', z) > Pen(D', x^*)$$

and thus x* is the fused value of D'.

In the preceding we have shown that selection of the fusion operator $F(a_1, a_2, ...a_n)$ as the value that minimizes Pen(D, x) always leads to a fusion operator that is

1. Idempotent
2. Commutative
3. Is bounded by

$$Min_i\, a_i \leq F(a_1, a_2, ...a_n) \leq Max_i\, [a_i]$$

4. Exhibits the self identity property

5.3. On Some Classes of Fusion Operators

In this section we shall consider some specific formulations for the local penalty function P and see what types of fusion functions they result in.

First we consider the case in which we incur a fixed penalty when we suggest a fused value different from an observed value. In this situation we formally express our local penalty function as

$$P(a_i, x) = 0 \qquad \text{if } x = a_i$$

$$P(a_i, x) = K \qquad \text{if } x \neq a_i.$$

Consider now any potential fused value x. In this case

$$Pen(D, x) = (n - n_x)K$$

where n_x is the number of times x appears in D. Thus we see that Pen (D, x) attains it minimal value for the value that appears most times in D. Hence in this case if n_x equals the number of times x appears in D then

$$F(a_1, a_2, ..., a_n) = \{x \mid \text{such that } n_x = \text{Max}_y\, n_y\}$$

We note this operator is the mode aggregation. Thus the mode fusion occurs when we assume a fixed penalty function. Let us look at some properties of this aggregation. First we have already established its idempotency as well its commutativity. We must now look and see if it is monotonic. The following example illustrated that this mode type aggregation is not monotonic.

Example: Consider

$$D = \{7, 7, 7, 3, 3, 2, 2\}$$

In this case the mode type fused value is 7. Consider now

$$D' = \{7, 7, 7, 3, 3, 3, 3\}$$

In this case we see that the fused value is 3, which is of course less then seven. However, we also see that if we denote the objects in D as a_i and those in D' as b_i then $b_i \geq a_i$. Hence monotonicity is not satisfied.

On the other hand we can show that the mode type aggregation exhibits the property of self identity. Assume that x is the mode of D, x is the element that appears most times in D. If we add another element to D equal to x then x still appears the most times in D and hence is still the mode.

The one very nice property associated with this type of aggregation is that it doesn't require the fused values to be numbers. As a matter of fact it doesn't ever require an ordering on the space from which the objects are draw. That is we can use this operator in non-numeric domains. Thus the objects to be fused can be any elements.

Is should also be noted that this type of aggregation doesn't always lead to a unique solution, for it can often be the case that more then one element has the maximum number of appearances. Thus it may require a second step to select among the optimal solutions.

A second class of local penalty functions are those of the form

$$P(a_i, x) = (a_i - x)^2.$$

In this case we must find x such that x minimizes

$$Pen(x, D) = \sum_{i=1}^{n} (a_i - x)^2.$$

This is a well known optimization problem and leads to the simple average as its fused value,

$$x = \frac{1}{n} \sum a_i.$$

It can be easily seen that this is monotonic. Thus can also be seen to satisfy the self identity property. Assume

$$a_{n+1} = \frac{1}{n} \sum_{i=1}^{n} a_i$$

then

$$x = \frac{1}{n+1} [\sum_{i=1}^{n} a_i + \frac{1}{n} \sum_{i=1}^{n} a_i]$$

$$x = (\frac{1}{n+1}) (\frac{1}{n}) (\sum_{i=1}^{n} n\, a_i + \sum_{i=1}^{n} a_i)$$

$$x = (\frac{1}{n+1}) (\frac{1}{n}) \sum_{i=1}^{n} (a_i)(n+1) = \frac{1}{n} \sum_{i=1}^{n} a_i$$

A very significant property of this type of aggregation is that it always leads to a unique answer. We also note this type of aggregation strictly requires numbers.

We can consider a generalization of this is domains in which our observations are non–negative. Consider the local penalty function of the form

$$P(a_i, x) = (a_i^{\alpha} - x^{\alpha})^2$$

where $\alpha \in [-\infty, \infty]$.

In this case

$$Pen(D, x) = \sum_{i=1}^{n} (a_i^{\alpha} - x^{\alpha})^2$$

Taking the derivative of this and setting it to zero we get

$$\frac{\partial}{\partial x} Pen(D, x) = - \sum_{i=1}^{n} 2(a_i^{\alpha} - x^{\alpha}) (\alpha) x^{\alpha - 1} = 0$$

From this we get

$$\sum_{i=1}^{n} (a_i^{\alpha} - x^{\alpha}) = 0$$

which gives us

$$x = (\frac{1}{n} \sum_{i=1}^{n} a_i^{\alpha})^{1/\alpha}$$

This is the so called generalized mean discussed by Pedrycz [Dyckhoff, Pedrycz, 1984].

A number of special cases are worth noting. First of course if $\alpha = 1$ we simply get the usual average. If $\alpha = -1$ then $(a_i, x) = P(\frac{1}{a_i} - \frac{1}{x})^2$ and as shown in [Dyckhoff, Pedrycz, 1984],.[Yager, Filev, 1994] this results in the harmonic mean

$$x = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \ldots \frac{1}{a_n}}.$$

In the case when $\alpha \rightarrow \infty$ this aggregation leads to [Dyckhoff, Pedrycz, 1984],.[Yager, Filev, 1994]

$$x = Max_i [a_i]$$

and when $\alpha \rightarrow -\infty$ this aggregation leads to [7, 8]

$$x = Min_i [a_i].$$

We now consider another generalization of the local penalty function $(a_i - x)^2$. Here we consider the function

$$P(a_i, x) = (a_i - x)^2 \wedge K,$$

where $\wedge$ is the minimal operator. In this case the penalty associated with any observation is bounded by a value K, which is the maximal local penalty that can be occurred from observation a_i. There exists an interesting semantics that can be associated with this formulation. We can view K as a penalty or price we pay for discounting the observation a_i. Thus we choose to not consider observation a_i if we find it better to pay this penalty. In this environment we must again find x to

minimize

$$\text{Pen}(D, x) = \sum_{i=1}^{n} P(a_i, x).$$

This problem can be structured as a mathematical programming problem.

$$\text{Min: } q_1 + q_2 + \ldots q_n$$

such that

$$q_i \geq (x - a_i)^2 - m\, z_{i1} \qquad \text{for } i = 1, \ldots, n$$

$$q_i \geq K - m\, z_{i2} \qquad \text{for } i = 1, \ldots, n$$

$$z_{i1} + z_{i2} = 1 \qquad \text{for } i = 1, \ldots, n$$

$$z_{i1} \text{ and } z_{i2} \text{ must be 0 or 1} \qquad \text{for } i = 1, \ldots, n$$

In the above m is some very large number that always guaranties that $(x - a_i)^2 - m$ and $K - m$ are negative.

We now go to another class of penalty functions. Here we consider the case in which the local penalty incurred is equal to the absolute difference between the observed value and the fused value,

$$P(a_i, x) = |a_i - x|.$$

It is easy to see that this satisfies our required conditions for a local penalty function. Consider now the overall penalty function

$$\text{Pen}(D, x) = \sum_{i=1}^{n} |a_i - x|$$

Let us assume that the a_i's are indexed such that $a_i \leq a_j$ for $i < j$, $a_1 \leq a_2 \leq a_3 \ldots \leq a_n$.

We now show that the minimal value of Pen(D, x) can't be less then a_1 or greater then a_n. Consider first finding the minimal value of Pen(D, x) for $x \leq a_1$. For any $x \leq a_1$ we have

$$\text{Pen}(D, x) = \sum_{i=1}^{n} (a_i - x)$$

The minimum of the above is obtained by making x as large as possible. However, since x is restricted to be less then or equal a_1 we see that the minimum occurs when $x = a_1$. Now consider the determination of the minimum value in the range $x \geq a_n$. For any $x \geq a_n$ we have

$$\text{Pen}(D, x) = \sum_{i=1}^{n} (x - a_i).$$

The minimal value of the above is obtained by making x as small as possible. However,

since x is restricted to be greater then or equal a_n we see that the minimum occurs when $x = a_n$.

Thus based on the above we have determined that the value of x that minimizes

$$Pen(D, x) = \sum_{i=1}^{n} |a_i - x|$$

must be in the interval

$$a_1 \leq x \leq a_n.$$

Thus we see that the fused value must always be between the maximal and minimal observation. We see therefore our problem becomes that of finding the optimal value lying between these extremes.

Consider now the case in which we only have one observation a_1. In this case

$$Pen(D, x) = |x - a_i|$$

and the minimum value of course occurs at $x = a_1$. Thus in this case our fused value occurs at $x = a_1$.

Now consider the situation in which we have two observations, a_1 and a_2, where have assumed $a_1 \leq a_2$. Since we have already shown that the minimum value occurs in the interval $[a_1, a_2]$ we can consider finding the value of x that minimizes

$$Pen(D, x) = (x - a_1) + (a_2 - x) = a_2 - a_1.$$

Thus in the interval $[a_1, a_2]$, where the minimal of Pen(D, x) must be, we see that Pen(D, x) is constant and any element $x \in [a_2, a_1]$ provides the minimal value. Thus the fused value is any element in this interval.

Before we proceed to the more general case we shall make an observation. Consider any function F(x). Assume we can express this function as

$$F(x) = F_1(x) + F_2(x) + ...F_q(x).$$

If there exists some value x^* which minimizes each of the $F_i(x)$ then this value must minimize the function F(x).

Consider now the penalty function

$$F(x) = \sum_{i=1}^{n} |a_i - x|$$

where again we have assumed the a_i's are indexed in increasing order of magnitude. We shall first

assume that n is odd, $n = 2q + 1$. We next rewrite F(x) as

$$F(x) = F_1(x) + F_2(x) + ...F_q(x) + F_{q+1}(x)$$

where for $i = 1, ...q$

$$F_i(x) = |a_i - x| + |a_{n-i+1} - x|$$

and

$$F_{q+1} = |a_{q+1} - x|.$$

Consider any term F_i where $i = 1, ...q$. We have already established that the minimum value of this term occurs at any point in the interval $[a_{n-i+1}, a_i]$. Let us denote $R_i = [a_{n-i+1}, a_i]$ for $i = 1, ..., q$. Because of assumed indexing we have this

$$R_1 \supseteq R_2 \supseteq R_3 \supseteq ...\supseteq R_q.$$

Thus these ranges form a nested collection of intervals. Let us now consider the problem of minimizing F(x). Since the minimal of $F_1(x)$ occurs in any point of the interval R_1 and the minimal of $F_2(x)$ occurs at any point in interval R_2, and $R_2 \subseteq R_1$, then the minimal of $F_1(x) + F_2(x)$ occurs in R_2. Furthermore, since the minimal of $F_3(x)$ occurs at any point in the interval R_3 and since $R_3 \subseteq R_2$ and the minimal of $F_1(x) + F_2(x)$ lies in the interval R_2 then the minimal of $F_1(x) + F_2(x) + F_3(x)$ occurs in the interval R_3. Continuing in this manner we see that the minimal of $F_1(x) + F_2(x) + ...F_q(x)$ occurs anywhere in the interval $R_q = [a_q, a_{q+2}]$. We must now consider F_{q+1}. We have already shown that the minimal of this occurs at $x = a_{q+1}$. Since $a_{q+1} \in [a_q, a_{q+2}]$ we see that the minimal of $F(x) = F_1(x) + ...F_q(x) + F_{q+1}(x)$ occurs at $x = a_{q+1}$. Since as we already noted the a_i's where indexed in increasing order we note that a_{q+1} is the median value. Thus using the penalty function $|a_i - x|$ we attain the median aggregation method.

In the case in which n is even, $n = 2q$, we see that we no longer need the term F_{q+1}. Thus in this case the optimal occurs in the interval R_q where

$$R_q = [a_q, a_{q+1}]$$

Thus again we get a median solution, that being the range between the two middle values.

We easily see that the median solution can be obtained by just ordering the observations and taking the middle value if the number of observations is odd or the range between the two middle

values if the number of observation is even.

It is important to note that to use this approach all we really need is an ordering on the observation space. It can be easily seen that the median aggregation is monotonic.

The value of the penalty function when using a medina aggregation can be easily calculated. First if n is even

$$Pen(D, x^*) = \sum_{i=1}^{q} F_i(x^*)$$

where x^* is the median. If n is odd

$$Pen(D, x^*) = \sum_{i=1}^{q} F_i(x^*) + F_{q+1}(x^*)$$

however since $F_{q+1}(x^*) = 0$ we get in either the odd or even case

$$Pen(D, x^*) = \sum_{i=1}^{q} F_i(x^*)$$

Consider any term

$$F_i(x^*) = |a_i - x^*| + |a_{n-i+1} - x^*|$$

Since for the median value x^* we have

$$a_i \leq x^* \leq a_{n-i+1}$$

we get

$$F_i(x^*) = a_{n-i+1} - a_i$$

and therefore

$$P(D, x^*) = \sum_{i=1}^{q} a_{n-i+1} - \sum_{i=1}^{q} a_i.$$

In particular we see that $\sum_{i=1}^{q} a_{n-i+1}$ is the sum of the values above the median and $\sum_{i=1}^{q} a_i$ is the sum of values below the median. Thus

$$Pen(D, x^*) = \text{Total above median} - \text{Total below median}$$

In this section we have considered three classes of local penalty function, constant, square of the difference and absolute difference. We have shown that they led respectively to the mode, mean and median fusion techniques.

3.4. Weighted Penalty Functions

In this section we assume that each of the observations have an associated weight function. This weight can be a measure of the importance or reliability of that sensor. Thus

$$\text{Pen}(D, x) = \sum_{i=1}^{n} w_i \, P(a_i, x).$$

If

$$P(a_i, x) = 0 \qquad a_i = x$$

$$P(a_i, x) = K \qquad a_i \neq x$$

then

$$\text{Pen}(D, x) = K \sum_{i=1}^{n} w_i - K \sum_{\substack{\text{for all } j \\ \text{s.t.} \\ a_j = x}} w_j$$

To obtain the optimal solution we proceed as follows:

1. Let Y be the set of distinct elements that appear in D, $Y = \{y_1, ..., y_q\}$

2. For each y in Y calculate

$$G(y) = \sum_{\substack{j \text{ s.t.} \\ a_j = y}} w_j$$

3. Select as fused value y^* such that

$$G(y^*) = \max_{y \in Y} G(y).$$

We note that this approach works even if the observations are drawn from a space X which has no structure.

Example: Let X = {red, blue, green, yellow}. Assume our observation collection is

D = {(0.6, red), (0.9, blue), (0.3, red), (0.1, red), (0.7, yellow), (0.2, red), (0.8, blue)}

here the numbers are the associated weight. The set of the observed values are {red, blue, yellow} We see that .

G(red) = 0.6 + 0.3 + 0.1 + 0.2 = 1.2

G(blue) = 0.9 +0 .8 = 1.7

G(yellow) = 0.7

Thus the fused value is blue.

The case in which

$$P(a_i, x) = (a_i - x)^2$$

leads to the following penalty function when weights are included.

$$Pen(D, x) = \sum_{i=1}^{n} w_i (a_i - x)^2$$

The value that optimizes this is the well known weighted average

$$x = \frac{\sum_{i=1}^{n} a_i w_i}{\sum_{i=1}^{n} w_i}.$$

Let us consider the modified version of this formulation where

$$P(a_i, x) = (a_i^{\alpha} - x^{\alpha})^2$$

where $\alpha \in r$ and the a_i are non-negative. In this case

$$Pen(D, x) = \sum_{i=1}^{n} w_i (a_i^{\alpha} - x^{\alpha})^2$$

Taking the derivative of this and setting it equal to zero we get

$$\sum_{i=1}^{n} 2w_i (a_i^{\alpha} - x^{\alpha}) (\alpha - 1) x = 0$$

For the solution to this we get

$$\sum_{i=1}^{n} w_i a_i^{\alpha} = x^{\alpha} \sum_{i=1}^{n} w_i$$

and hence

$$x = \left(\frac{\sum_{i=1}^{n} w_i a_i^{\alpha}}{\sum_{i=1}^{n} w_i}\right)^{1/\alpha}$$

We now consider the case in which our local penalty function is

$$P(a_i, x) = |a_i - x|$$

In the previous section we showed this form of local penalty led to a median aggregation method. Here we shall consider the weighted penalty function

$$Pen(a_i, x) = \sum_{i=1}^{n} w_i |a_i - x|$$

where $w_i \geq 0$. We must again find x that minimizes this sum. Since a normalization of the weights will not affect this solution we shall without loss of generality assume the weights are normalized,

$$\sum_{i=1}^{n} w_i = 1.$$

We first note that if we have only one observation, a_1, the optimal value occurs at $x = a_1$. Hence in this case a_1 is the fused value.

Consider the case in which we have two observations. $a_1 \leq a_2$. Initially we shall assume that $w_1 = w_2 = w$. We note that this requires $w = .5$. In this case since the minimal must be in the interval $[a_1, a_2]$ we have

$$\text{Pen}(a_i, x) = w(x - a_i) + w(a_2 - x)$$

$$= w(a_2 - a_1)$$

Since this is the same for all $x \in [a_1, a_2]$ our fused value is any point in the interval $[a_1, a_2]$. Now we consider the case in which one of the components has a larger weight, without loss of generality we shall assume $w_1 > w_2$. In this case

$$\text{Pen}(D, x) = w_1 (x - a_1) + w_2(a_2 - x)$$

Since $w_1 > w_2$ we can express

$$w_1 = w_2 + \Delta$$

where $\Delta > 0$. Using this we get

$$\text{Pen}(D, x) = \Delta (x - a_1) + w_2 (x - a_1) + w_2 (a_2 - x)$$

$$\text{Pen}(D, x) = \Delta (x - a_1) + w_2 (a_2 - a_1).$$

We see therefore that the optimal value, fused value, occurs at $x = a_1$. We should note that it can easily be shown that if $w_2 > w_1$ the fused value occurs at $x = a_2$.

Let us now consider the general case where

$$\text{Pen}(D, x) = \sum_{i=1}^{n} w_i |a_i - x|$$

where it is assumed that the a_i's are indexed such that $a_i \leq a_j$ if $i < j$ and of course the weights have been normalized. Let K be any value such that we can express any of the w_i in the form

$$W_i = m_i K$$

where m_i is a positive integer. For example, if $w_1 = 0.3$, $w_2 = 0.15$, $w_3 = 0.41$, $w_4 = 0.14$ then with $K = .01$ we have $m_1 = 30, m_2 = 15$, $m_3 = 41$ and $m_4 = 14$.

Using this substitution we get

$$Pen(D, x) = \sum_{i=1}^{n} m_i K |a_i - x|$$

To find the optimal value we can consider the function

$$Pen(D, x) = \sum_{i=1}^{n} m_i |a_i - x|$$

where each of the m_i are positive integers. Further let us denote $m = \sum_{i=1}^{n} m_i$.

Let us introduce a new set of variables b_j, where $j = 1$ to m. We denote the b_j's as follows:

$$b_j = a_i \qquad \text{for } j = S_{i-1} + 1 \text{ to } S_i$$

where

$$S_0 = 0$$

$$S_i = S_{i-1} + m_i$$

Consider now the term

$$\sum_{j=1}^{m} |b_j - x|$$

this is exactly equal to

$$\sum_{i=1}^{n} m_i |a_i - x|.$$

and hence we can consider as our penalty function

$$Pen(D, x) = \sum_{j=1}^{m} |b_j - x|.$$

We note that this is a non-weighted form and exactly the type of aggregation we considered in the earlier section. In particular the solution to this is the median value of the b_j.

We shall try to get this solution directly in terms of the original a_i's and the associated weights. We shall first consider the case where m is odd, $m = 2q + 1$. In this case the median aggregation value is b_{q+1}. Let us now try to find the a_i value associated with b_{q+1}. The simplest way to do this is to start counting the m_i values. In particular we see that $b_{q+1} = a_i^*$ for the value of i^* such that

$$S_{i^*-1} \leq q$$

$$S_{i^*} \geq q + 1$$

Furthermore, we can express the solution in terms of the original weights. The optimal solution a_i^* occurs for the value of i^* such that

$$K S_{i^*-1} \leq q K$$

$$K S_{i^*} \geq q K + K.$$

Consider a term

$$KS_i = \sum_{j=1}^{i} K m_j = \sum_{j=1}^{i} w_j$$

Thus if we denote $T_i = \sum_{j=1}^{i} w_i$, the sum of the first i weights, then our rule for finding i^* is

$$T_{i^*-1} \leq qK$$

$$T_{i^*} \geq qK + K$$

Furthermore since K m = 1 and m = 2q + 1 then $q = \frac{m-1}{2}$ and $Kq = \frac{mK-K}{2} = \frac{1}{2} - \frac{K}{2}$.

Therefore we see i^* occurs for

$$T_{i^*-1} \leq \frac{1}{2} - \frac{K}{2} < 0.5$$

$$T_{i^*} \geq \frac{1}{2} - \frac{K}{2} + K > 0.5$$

Thus the weighted median occurs for the i value where the sum of the ordered weights go from being less then 0.5 to being greater then 0.5.

We now consider the case when m is even, m = 2q. In this case the median value is the interval $[b_q, b_{q+1}]$. Let us now try to find the a_i values associated with b_q and b_{q+1}. Again we shall use a counting of the m_i values. In particular we se that $b_q = a_{i^*}$ for the value of i^* such that

$$S_{i^*-1} \leq q - 1$$

$$S_{i^*} \geq q$$

and $b_{q+1} = a_i{}^{**}$ for the value of i^{**} such that

$$S_{i^{**}-1} \leq q$$

$$S_{i^{**}} \geq q + 1$$

Multiplying each of the above by K we get

$$T_{i^*-1} \leq qK - K$$

$$T_{i^*} \geq qK$$

and

$$T_{i^{**}-1} \leq qK$$

$$T_{i^{**}} \geq qK + K.$$

Since $m = 2q$ and $mk = 1$ we get $qK = \frac{mK}{2} = 0.5$ and therefore

$$T_{i^*-1} \leq 0.5 - K < 0.5$$

$$T_{i^*} \geq 0.5$$

and

$$T_{i^{**}-1} \leq 0.5$$

$$T_{i^{**}} \geq 0.5 + K > 0.5$$

Two cases must be distinguished here. In the first case there exists a value r such that $T_r = \sum_{j=1}^{r} w_j = 0.5$. In this case $bq = a_r$ and $b_{q+1} = a_{r+1}$. In the second case there is no value r for which $T_r = \frac{1}{2}$. In this case $b_q = b_{q+1}$ and it occurs at the value a_r for which

$$T_{r-1} < 0.5$$

$$T_r > 0.5.$$

In the following we shall summarize the results of the preceding to provide an approach to weighted median aggregation. Assume we have a collection of values to be fused

$$<(a_i, w_i)>$$

where $i = 1, ..., n$. In the above a_i is the value and w_i is the weight associated with that value. We shall assume the weights are normalized,

$$w_i \in [0, 1]$$

$$\sum_{i=1}^{n} w_i = 1.$$

Furthermore, we assume that the a_i's have been indexed such that

$$a_i \leq a_j \qquad \text{if } i < j.$$

In the following we shall let

$$T_j = \sum_{i=1}^{j} w_i.$$

Thus T_j is the sum of the weights of the first j ordered observations. The following algorithm finds the weighted median

(1) Initiate $j = 1$

(2) Calculate T_j

(3) If $T_j \geq 0.5$ go to step 5

(4) Set $j = j + 1$ and go to step 2

(5) If $T_j > 0.5$ set $x = a_j$

(6) If $T_j = 0.5$ set $x \in [a_j, a_{j+1}]$

The following example illustrates the procedure

Example: Find the weighted median of

$D = <((0.1, 3), (0.2, 5), (0.25, 6), (0.1, 7), (0.1, 9), (0.15, 13), (0.1, 19))>$

a_i	w_i	T_i	
3	.01	0.1	
5	0.2	0.3	
6	0.25	0.55	←
7	0.1	0.65	
9	0.1	0.75	
13	0.15	0.9	
19	0.1	1.0	

Thus the weighted median occurs at 6.

It should be noted that this procedure only requires an ordering on the values to be aggregated.

5.5. Possibility of No Solution

In the preceding we have made the implicit assumption that we must always arrive at a solution. In this section we shall investigate the possibility of saying that no solution exists. In the following we use $x = \emptyset$ to indicate that no fused value is provided.

Consider first the following penalty function

$$P(x, a) = K_1 > 0 \quad \text{if } x = \emptyset \text{ (no solution)}$$

$$P(x, a) = 0 \quad \text{if } x = a$$

$$P(x, a) = K_2 > 0 \quad \text{if } x = b \neq a.$$

In the above K_1 is the price we must pay for providing no solution and K_2 is the price we

must pay when providing a solution that disagrees with an observation.

Again our objection is to find s so as to minimize

$$\text{Pen}(D, x) = \sum_{i=1}^{n} P(x, a_i).$$

If we decide to provide a solution, then we have shown that the minimal value of x occurs at the value that occurs most times in the observation set. Let n_x be the number of times the most often appearing value in D appears. In this case

$$\text{Pen}(D, x) = K_2 (n - n_x).$$

On the other hand if we provide no solution

$$\text{Pen}(D, \emptyset) = K_1 n.$$

We provide some solution if

$$K_1 n > K_2 (n - n_x)$$

$$1 - \frac{n_x}{n} < \frac{K_1}{K_2}$$

We note that $1 - \frac{n_x}{n} \leq 1$. From this we see that if $K_1 > K_2$ then we always suggest a solution, the mode. Thus if the penalty of not providing a solution is greater then the penalty of providing a disagreeable solution we provide some solution. We also note that if $n_x = n$ then $1 - \frac{n_x}{n} = 0$ and hence we always provide a solution. Thus no matter what our penalties are if all the observations are the same, we get a solution. Thus this approach exhibits idempotency.

We also note that if K_2 gets very large, we have a heavy penalty for providing a disagreeable solution, then we usually provide no solution, unless all observations are the same. In particular, we note that if K_2 gets very large we essentially take the intersection of the observations. Thus the intersection is the effective fusion method when the cost of disagreements is large.

We can consider a fuzzy generalization of the above. Let F be a fuzzy subset of the unit interval such that

$$F(1) = 1$$

$$F(y) \geq F(z) \qquad \text{if } y \geq z$$

We then calculate

$$\mu = F(\frac{n_x}{n}).$$

This would be used to indicate the degree to which x can be considered an appropriate aggregation of the observations in D.

We now consider the penalty function

$$P(x, a) = K^2 > 0 \qquad \text{if } x = \emptyset$$

$$P(x, a) = (x - a)^2 \qquad \text{if } x \neq \emptyset.$$

In this case if we decided to select a solution its value is

$$x = \frac{\sum_{i=1}^{n} a_i}{n}$$

and in this case the penalty is

$$\text{Pen}(D, x) = \sum_{i=1}^{n} (x - a_i)^2.$$

If we decide to provide no solution then

$$\text{Pen}(D, \emptyset) = K^2 n.$$

We then provide the solution x, if

$$K^2 n > \sum_{i=1}^{n} (x - a_i)^2$$

$$K^2 > \frac{1}{n} \sum_{i=1}^{n} (x - a_i)^2$$

$$K > \left(\frac{1}{n} \sum_{i=1}^{n} (x - a_i)^2\right)^{\frac{1}{2}}$$

Thus we provide a solution if the standard deviation is less then the penalty of providing no solution.

A fuzzy extension of the above idea can be had. Let us denote

$$s = \left(\frac{1}{n} \sum_{i=1}^{n} (x - a_i)^2\right)^{\frac{1}{2}}$$

as the standard deviation. Let G be a fuzzy subset of the non-negative real line such that

$$G(0) = 1$$

and

$$G(y) \leq G(z) \qquad \text{if } y < z.$$

We then calculate

$$\mu = G(s)$$

as the degree to which there exists a fused solution to our data set.

We now consider the penalty function

$$P(x, a) = K \qquad \text{if } x = \emptyset$$

$$P(x, a) = |x - a| \qquad \text{if } x \neq \emptyset.$$

In this case if we decide not to select a solution our penalty value is

$$Pen(D, \emptyset) = Kn$$

If we decide to have a solution, the median, then if $n = 2q$ or $n = 2q + 1$ then

$$Pen(D, x) = \sum_{i=1}^{q} a_{n+1-i} - \sum_{i=1}^{q} a_i$$

We then decide to provide a solution if

$$Kn > \sum_{i=1}^{q} (a_{n+1-i} - a_i)$$

$$K > \frac{1}{n} \sum_{i=1}^{q} (a_{n+1-i} - a_i)$$

A fuzzification of this approach can also be had. We define the disparity as

$$Disp(D) = \frac{1}{n} \sum_{i=1}^{q} (a_{n+1-i} - a_i).$$

Let G be a fuzzy subset defined on the non-negative real numbers such that

$$G(0) = 1$$

$$G(x) \geq G(y) \qquad \text{if } x > y.$$

Then

$$\mu = G(Disp(D))$$

indicates the degree to which the median provides an acceptable solution.

6. Fuzzy Systems and Neural Networks.

6.1 Simulation of Neural Networks Using Fuzzy Systems.

In describing neural networks we use the standard notation [Rich, Knight, 1991]. A neural network is represented by a graph of nodes and edges. Nodes are divided into sets of input, output, and hidden nodes. Input nodes constitute the input layer, output nodes constitute the output layer, and hidden nodes constitute the hidden layer. Each node of the hidden layer is connected to every node of the input layer by an edge, and, similarly, each node of the output layer is connected to every node of the hidden layer by an edge. A level of activation of a node in a hidden layer of a neural network is computed as a sigmoid activation function $S(t) = (1 + \exp(-t))^{-1}$) (appendix, fig 6.1) of the sum of weighed inputs:

(6.1) $\quad h_i = (1 + \exp(-\Sigma_j w1_{ij} \cdot x_j))^{-1} = S(\Sigma_j w1_{ij} \cdot x_j).$

A similar formula can be written for output units, whose inputs are units in the hidden layer:

(6.2) $\quad o_k = (1 + \exp(-\Sigma_j w2_{kj} \cdot h_j))^{-1} = S(\Sigma_j w2_{kj} \cdot h_j).$

When we compare fuzzy systems with neural networks we see two crucial differences. The first one relates to the computation of the output functions: in a neural net, it is a sigmoid function of a weighted sum of inputs at each level, whereas in a fuzzy system it is a conjunction of inputs in each rule at the first level and the aggregation of rules at the second level. The second major difference lies in the treatment of variables. In neural networks each variable is treated uniformly no matter what the range of the variable. In a fuzzy system each rule depends on the range of a variable (if x_1 is S_i^j ...).

To simulate neural networks via fuzzy systems we identify each unit at the second (hidden) level with the output of one rule of the fuzzy system. While there are a few ways to do this we will follow [Sugeno & Kang, 1988], where fuzzy system model rules are built in following form:

R_i: if x_1 is S_1^i and if x_2 is S_2^i and ... x_p is S_p^i then

(6.3) $\quad v^i = a_0^i + a_1^i x_1 + a_2^i x_2 + \ldots + a_p^i x_p,$

where x_j for $j=1,\ldots,p$ are inputs, $\{a_j^i\}$ is a parameter set and S_j^i is a fuzzy linguistic qualifier. We modify formula (6.3) to

R_i: if x_1 is S_1^i and if x_2 is S_2^i and ... x_p is S_p^i then

(6.4) $\quad v^i = S(a_0^i + a_1^i x_1 + a_2^i x_2 + \ldots + a_p^i x_p)$,

where S is a sigmoid function. We use the sigmoid function to imitate the neural network as closely as possible. This formula can simulate hidden units as in (6.1) with $a_i^i = w1_{ij}$ and $v^i = h_i$.

The above representation eliminates the first difference between output functions in neural networks at the hidden level and fuzzy systems.

In [Sugeno & Kang, 1988] output is inferred by the weighted average of v^i:

(6.5) $\quad v^0 = \Sigma_i v^i \cdot w_i / \Sigma_i w_i$.

If normalized weight is defined as

(6.6) $\quad \delta_i = w_i / \Sigma_i w_i$, $i = 1,\ldots,N$

then output can be rewritten as

(6.7) $\quad v^0 = \Sigma_i v^i \cdot \delta_i$, $i = 1,\ldots,N$

In a neural network the output of the unit k is computed in (6.2) as

$o_k = (1 + \exp(-\Sigma_j w2_{kj} \cdot h_j))^{-1} = S(\Sigma_j w2_{kj} \cdot h_j)$. If we change the formula in (6.7) to

(6.8) $\quad v^0 = S(\Sigma_i v^i \cdot \delta_i)$, $i = 1,\ldots,N$

then we can simulate the output of the neural network. The latter formula computes an output of the fuzzy system. The sigmoid function (appendix, fig. 6.1) can be interpreted as the rule: if an argument is much greater than zero then the result is equal to one.

Therefore, in this model an output can be interpreted as a rule:

(*) if the weighted sum^0 is much greater than zero, then output is 1.

In neural networks there are usually more than one output unit. Therefore, to simulate the whole neural network we have to build corresponding rules for each output unit in the neural network. In this case, the formula corresponding to the output function in (6.2) is

(6.9) $\quad v^k = S(\Sigma_j v^j \cdot \delta_j^k)$,

where $\delta_j^k = w2_{kj}$, $v^j = h_j$, and $v^k = o_k$.

The rule in fuzzy systems corresponding to this formula is:

(**) if the weighted sum^k is much greater than zero then output is 1.

6.2. Building Fuzzy Sets.

Formula (6.3) simulates the hidden units in neural networks. But the $x_1, x_2, \ldots, x_p$ themselves don't form fuzzy sets. There is no rule like the ones existing in fuzzy systems of the form: if x_1 is A_1^j and ... and x_p is A_p^j then y is B^j. We want to simulate neural networks via fuzzy systems. However, in these two frameworks variables are treated differently from variables in fuzzy systems.

In formula (6.3) we sum not over variables themselves but over weighted variables. But a weighted variable $a_j^i x_j$ can be interpreted as degree of membership of x_j in a fuzzy linguistic qualifier: if x_j is S_j^i (appendix, fig. 6.2).

In this picture $\text{cotan}(\alpha) = a_j^i$ (this includes negative weights as well).

For each variable x_j there are different set of linguistic qualifiers S_j^i; each of them is defined by parameter a_j^i. If a_j^1 is much larger than a_j^2 then S_j^1 can be described as Positive Large and S_j^2 as Positive Medium (or, alternatively, Positive Medium and Positive Small) (appendix, fig. 6.3 and fig. 6.4)

Parameters can be negative as well. Suppose that a_1^i is less than zero. In this case we have to transform $a_0^i + a_1^i x_1 + \ldots + a_p^i x_p$

to $(a_0^i - b_1^i) + (b_1^i + a_1^i x_1) + \ldots + a_p^i x_p$, where $(b_1^i + a_1^i x_1) > 0$

for **all relevant** values of x_1. The fuzzy set S_j^1 (Negative Medium) corresponds to the expression $b_1^i + a_1^i x_1$ (appendix, fig. 6.5)

Let us denote degree of membership $a_j^i x_j$ as g_j^i. Then (6.4) can be rewritten as

R_i: if x_1 is S_1^i and if x_2 is S_2^i and ... x_p is S_p^i then

(6.10) $\quad v^i = S(g_0^i + g_1^i + g_2^i + \ldots + g_p^i)$.

These representations eliminate the second major difference in output functions of neural networks and fuzzy systems. Using this model, fuzzy systems are able to simulate the work of neural networks. As an example, consider the neural network with input, output, and hidden layer with weight between input unit i and where hidden unit j is $w1_{ij}$, and weight between hidden unit k and output unit l is $w2_{kl}$ (appendix, fig. 6.6).

In this neural network $w1_{11} = 0.3$, $w1_{12} = 0.8$, $w1_{21} = -0.2$, $w1_{22} = 0.5$, $w1_{31} = 0.1$, $w1_{32} = 0.9$, $w2_{11} = -0.6$, $w2_{12} = 0.7$, $w2_{13} = 0.5$, $w2_{21} = 0.0$, $w2_{22} = 0.6$, $w2_{23} = -0.3$.

The fuzzy system corresponding to this neural network has the following rules (P -

positive, N - negative, S - small, B-big, M -medium):

if x_1 is PS & x_2 is NS & $x3_2$ is PS then h_1 is $S(g_0^1 + g_1^1 + g_2^1 + g_2^1)$

if x_1 is NS & x_2 is PM & $x3_2$ is PB then h_2 is $S(g_0^2 + g_1^2 + g_2^2 + g_2^2)$

(appendix, fig. 6.7 and fig. 6.8)

The aggregation functions for the first, second and third output are $y^1 = S(-0.6 \cdot h_1)$, $y^2 = S(0.7 \cdot h_1 + 0.6 \cdot h_2)$, $y^3 = S(0.5 \cdot h_1 - 0.3 \cdot h_2)$.

7. Parameters of Fuzzy Sets.

Fuzzy sets are sets primarily defined through predicates. For each object (which does belong to the domain of the predicate), we can say to what extent we can apply the predicate to this object. In other words, we order objects according to the definition of the predicate. Thus, **fuzzy sets are ordered sets** [Negoita, 1986]. The standard example is the set of tall people. Here the set of people is ordered according to the definition of the predicate 'tall' (people which are not tall, probably tall, possibly tall, not very tall, tall, very tall and so on). Thus, the above proposition should be the basis of the theoretical formalization of fuzzy sets.

Sets are ordered according to functions which assign to each object of a set a value of membership in each linguistic set (like positive big (PB), positive small (PS), around zero (AZ), and so on). Linguistic sets can be approximated by different basis-splines [Harris, 1994]. In this work we chose most commonly used triangle basis splines (appendix, fig. 7.1).

Triangle basis splines are defined via parameters $<k,t>$, where $k = \text{cotan}(\alpha)$. The function which assigns membership for object x of the set S in the linguistic set characterized by parameters $<k_i,t_i>$ is $(1 + \text{sgn}(k_{ij}x_i - t_{ij}))\times(1 - \text{sgn}(k_{ij}x_i - t_{ij} - 1))(k_{ij}x_i - t_{ij})/4 + + (1 + \text{sgn}(2 + t_{ij} - k_{ij}x_i))(1 - \text{sgn}(1 + t_{ij} - k_{ij}x_i))(2 + t_{ij} - k_{ij}x_i)/4$

Here, sgn is the signum function which is equal to -1, 0, or 1 depending on whether its argument is more than, equal to, or greater than 0.

Suppose that we have the fuzzy system set "speed of car" which depends on the angle of the road and friction.

If angle of the road is B and friction is S then the speed is AZ

If angle of the road is B and friction is M then the speed is L.

If angle of the road is B and friction is B then the speed is L.

If angle of the road is M and friction is S then the speed is L.

If angle of the road is M and friction is M then the speed is L.

If angle of the road is M and friction is B then the speed is M.

If angle of the road is S and friction is S then the speed is M.

If angle of the road is S and friction is M then the speed is H.

If angle of the road is S and friction is B then the speed is VH.

Fuzzy sets corresponding to angle of road, friction and speed are shown in fig. 7.2 in appendix.

Under these parameters, the angle 10 and friction 0.7 correspond to the speed 28.4.

7.1 Adjustments of Parameters in Fuzzy Systems.

As a rule, parameters are given by experts. But, as we have seen, the output depends on the parameters (and the shape) of the spline. It has been noted [Kosko, 1992] that neural nets and fuzzy systems are similar in their functions. In neural networks, the main problem is to adjust weights between perceptrons of different levels in such a way that the output of a neural network would be as close as possible to known output.

In fuzzy systems, parameters of a spline play a role similar to the weights in a neural network. The difference lies in the fact that in fuzzy systems these parameters are given by experts or assigned beforehand. But what if linguistic variables (like negative medium) are not totally adequate to describe a given process properly? Perhaps some other linguistic variable (like negative medium-small) would be more appropriate, or already given variables should have slightly different parameters. In the above example, if we change fuzzy set which defines medium angle of the road as shown in fig. 7.3 in appendix we get the speed 27.6.

In general, parameters, which define splines, can be considered as variables. Suppose that we want to build the fuzzy system which simulates the behavior of some controller, and we know outputs for a number of inputs. The original parameters of the fuzzy system are given by experts. Then we can adjust the parameters of the splines in a way similar to adjusting weights in neural networks.

The adjustment of parameters of splines can be written as a two-place function $f(<k,t>) = <k',t'>$, where k',t' are not necessarily all different from k,t. Because the rules of fuzzy systems are written as 'if x_1 is A_i, and if x_2 is B_j, and ...., and x_n is C_k then y is D_l', antecedents of each rule can be written as an n-tuple $<A_i,B_j,\ldots,C_m>$. As each of $A_i,B_j\ldots$ is a spline, the antecedent of each rule can also be written as $<k_i,t_i> \times <k_j,t_j> \times \ldots \times <k_m,t_m>$. Then, adjustment of parameters is a function $f(<k_i,t_i> \times <k_j,t_j> \times \ldots \times <k_m,t_m>) = <k_i',t_i'> \times <k_j',t_j'> \times \ldots \times <k_m',t_m'>$.

The adjustment of parameters even in one rule affects the output of the whole fuzzy system. So, the arguments of a parameter-adjusting function should be antecedents of all the rules of a fuzzy system, and, correspondingly, its range is also all the rules of a fuzzy system with p rules:

$$f(<k_1^1,t_1^1> \times <k_1^2,t_1^2> \times \ldots \times <k_1^n,t_1^1> \times <k_2^1,t_2^1> \times <k_2^2,t_2^2> \times \ldots \times <k_2^n,t_2^1> \times$$
$$\times \ldots \times <k_p^1,t_p^1> \times <k_p^2,t_p^2> \times \ldots \times <k_p^n,t_p^1> =$$
$$<k_1^{1'},t_1^{1'}> \times <k_1^{2'},t_1^{2'}> \times \ldots \times <k_1^{n'},t_1^{1'}> \times <k_2^{1'},t_2^{1'}> \times <k_2^{2'},t_2^{2'}> \times \ldots \times$$
$$\times <k_2^{n'},t_2^{1'}> \times \ldots \times <k_p^{1'},t_p^{1'}> \times <k_p^{2'},t_p^{2'}> \times \ldots \times <k_p^{n'},t_p^{n'}>.$$

If we denote the domain and range of this function as P and P' (P stands for parameters) respectively, then the above expression can be written as $f(P) = P'$. Henceforth, we'll use this notation.

We need to construct rules of fuzzy systems in such a way so that it provides the desired output for an arbitrary input. Similarly, in neural networks, adjustment of weights between perceptrons on different levels are made to minimize the difference between the output of the neural network and the control output.

In neural networks, weights converge to the optimal weights. If dU(W) is the gradient of difference between the output of the network and control output, the change of weights dW in the neural network is proportional to dU and its direction is opposite to dU. In a fuzzy system, parameters k_i,t_i play the role similar to the role of weights in a neural network, and the function $f(P) = P'$ should be made proportional to differences between the output of a fuzzy system and the desired output.

To understand this mechanism better the proper formalization of fuzzy systems is helpful.

7.2. Formalization of Fuzzy Systems.

To formalize the above structure, we form a category where the objects of the category are all antecedents of rules (in our terminology - P and P'), and adjustment functions between them ($f(P) = P'$) are the morphisms.

For each input in a fuzzy system with given rules, the corresponding output is the defuzzified value of the control variable. If the input is values of variables $x_1,x_2,\ldots,x_n$ and the output is the value of variable y, we can write this input-output as a function

$g(x_1,x_2,\ldots,x_n)=y$. These input-output functions form objects in another category - the category of fuzzy systems. Morphisms in this category are functions y--->y' such that the triangle in fig 7.4 in appendix commutes.

What does this function g depends on? Obviously it depends on the choice of parameters, which define membership of fuzzy variables. Thus, we can form a functor **F** from the category of fuzzy sets to the category of fuzzy systems:

$\mathbf{F}(P)$----->$(g<x_1,x_2,\ldots,x_n>$ ---> y).

The diagram for morphisms is shown in fig. 7.5 in appendix.

What else affects an output of a fuzzy system? First, the way variables represented by fuzzy sets are combined in each rule. We have a rule 'if x_1 is A_i, and if x_2 is B_j, and ..... and x_n is C_k then y is D_l'. There are a number of ways to evaluate the conjuction "and". In the above example, the membership of variable y was evaluated as $\min(m(x_1),m(x_2))$. But min is only one possible way of defining conjunction. It can be any operator which satisfies the following four conditions: 1) it should be symmetric; 2) it should be associative (these two condition provide that output doesn't depend on the order of variables); 3) it should be monotonic on any argument; 4) it should equal 0 if one of the arguments equals 0. It can be min, algebraic product, logarithmic product, bounded product, and so on.

Secondly, an output depends on the way outcomes of rules are aggregated. In the above example, an aggregation is defined as the maximum. In general there are a number of ways to define an aggregation (or disjunction). An aggregation operator should satisfy the following conditions: 1) it should be symmetric; 2) it should be associative (these two condition provide that output doesn't depend on an order of variables); 3) it should be monotonic on any argument; 4) it should equal 1 if one of arguments equals 1. It can be max, algebraic sum, p-bounded sum, logarithmic sum, and so on.

If, for example, we chose product instead of minimum operator we will get the speed 26.3.

The conjunction operator and aggregation operator determine the input-output relation for an arbitrary fuzzy system with defined fuzzy variables and defined rules. In the above diagram, their choice defines functor **F**. When we chose another conjunction and/or aggregation operator in terms of formal theory, it means that the diagram should be

constructed with another functor **G** instead of **F**. These functors form a category of their own where they form objects, and morphisms between them are natural transformations (appendix, fig.7.6).

Here $t(P) = P$, $t(<x_1,x_2,\ldots,x_n> \dashrightarrow y) = <x_1,x_2,\ldots,x_n> \dashrightarrow t(y)$ and $t(\mathbf{F}) = \mathbf{G}$.

Applying natural transformations to a category describing a fuzzy system, we shift from a system characterized by the one sample of conjunction and aggregation operators to a system with the same linguistic sets with another sample of conjunction and aggregation operators. Thus, we possess another tool to influence input-output function in such a way that better conforms to relevant data.

Originally fuzzy systems were used to find a way to deal with phenomena whose description is based on predicates. Herein lies the origin of the notion of a fuzzy set, a set which is defined through possession of a property (say, the property of being tall), but where we are not sure to what extent a given object has this property. Thus, most fuzzy systems are based on experts: what fuzzy sets constitute antecedents and consequents of rules. But why we are so sure that experts pick up the most suitable choices for fuzzy system antecedents and consequents? Maybe a slightly different fuzzy sets will work better? What experts in fact provide is that in general we wouldn't fall in some local minima, i.e. a situation where the input-output function doesn't necessarily give output sufficiently close to that required, but any change in the description of any one of the fuzzy sets increases the error. So, if it is possible to compare the output of a fuzzy system with actual in a number of control examples, we have to apply a procedure similar to the backpropagation technique in neural networks, i.e. adjust parameters defining fuzzy sets to minimize an error.

There is another issue. How should we compute conjunction in each rule and how should we make an aggregation? There is no direct answer to this question, but we should consider these two operations as one more level of the control. Most fuzzy systems apply minimum or product for conjunction, and maximum for aggregation. While these systems are efficient, they are not necessarily useful if we want, for example, that one set of rules have more priority than another. This property is important when we want to take into account "riskiness" of some combinations.

7.3. Estimation of Parameters of Antecedents.

Fuzzy systems have advantages in comparison to neural networks. Neural networks are "black boxes"; we don't know the rules which govern their behavior. In contrast, fuzzy systems are transparent: we are able to tell in what way particular output is related to given input. Rules of a fuzzy system are written in symbolic form, that is, the form people use to think. The knowledge of a neural network is an impenetrable mass of connection weights. It is difficult to explain its reasoning.

The above advantage is related to another one. Experts usually express their knowledge as rules, and don't necessarily have input-output pairs required to train a neural network. Fuzzy systems can easily incorporate an expert's knowledge, if it is given as rules, whereas a similar problem for a neural network presents difficulties.

The building of fuzzy system consists of two stages: initialization and adjustment. At the first step we have to take into account that each fuzzy set and each rule in a fuzzy system should have meaningful interpretation. This requirement means that linguistic variables used for the description of fuzzy sets be ones most commonly used in standard descriptions (like very tall, unlikely tall, not tall, or big angle (of the road), small angle, and so on). That provides for transparency of and tractability of a fuzzy system.

Therefore, initial parameters defining fuzzy sets should satisfy the above requirement. Initialization of fuzzy sets should be done by experts. They should use descriptions in their field of expertise to construct fuzzy sets. Only then is it possible to go to the second stage - adjustments.

Adjustments can be made in two steps. At the first step, adjustments are made in parameters which define fuzzy sets in antecedents of rules. At the second step, adjustments are made in parameters which define fuzzy sets in consequents. Suppose that conjunction in a fuzzy system is calculated as the *min* function, and aggregation (or defuzzification) as *max*. In short, we use a minimax procedure. In this case, rule j in a fuzzy system is $min_i(h_{ij}(x_i))$, where $h_{ij}(x_i)$ is a function defining degree of membership of fuzzy variable x_i. In triangular splines, provided that they are symmetric, $h_{ij}(x_i)$ is equal to $(1 + \mathrm{sgn}(k_{ij}x_i - t_{ij})) \times (1 - \mathrm{sgn}(k_{ij}x_i - t_{ij} - 1))(k_{ij}x_i - t_{ij})/4 + + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_i))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_i))(2 + t_{ij} - k_{ij}x_i)/4$. where k_{ij} and t_{ij} are parameters defining splines - corresponding to angle and shift of the line.

For an input $x_1,\ldots, x_n$, the output d is the value of the consequent fuzzy set, associated with the rule which gives maximal membership value, multiplied by this membership value. It is computed as $\Sigma_j r_j \times max_j(min_i(h_{ij}(x_i))) \times (1 + \mathrm{sgn}(min_i(h_{ij}(x_i)) - max_j(min_i(h_{ij}(x_i)))))$. The difference between this output and known result y is $(y - d) = (y - \Sigma_j r_j \times max_j(min_i(h_{ij}(x_i))) \times (1 + \mathrm{sgn}(min_i(h_{ij}(x_i)) - max_j(min_i(h_{ij}(x_i)))))$. We'll try to minimize this difference based only on this sample; the resulting fuzzy system will be perfectly fit just for this particular input-output pair (we can build a fuzzy system with one rule: if inputs are equal to given values $x_1,\ldots,x_n$ then the result is equal to given y). Therefore, first, the number of input-output pairs should be much greater than the number of rules, and second, input values for each variable x_i should cover all of its range. If there are k inputs $\{x_{11},\ldots,x_{n1}\},\ldots,\{x_{1k},\ldots, x_{nk}\}$ the corresponding output of the fuzzy system is $d_1,\ldots,d_k$. Known outputs for above inputs are $y_1,\ldots, y_k$, and for each input-output pair l the difference (or error) between the output of the fuzzy system and known output is equal to $e_l = (y_l - d_l) = (y_l - \Sigma_j(r_j \times max_j(min_i(h_{ij}(x_{il}))) \times (1 + \mathrm{sgn}(min_i(h_{ij}(x_i)) - max_j(min_i(h_{ij}(x_{il}))))))$. For m pairs, errors sum to $\Sigma_l^m e_l$.

We want to find parameters k_{ij} and t_{ij} that minimize this difference. To achieve this we apply a variation of the least-mean-square algorithm. In this algorithm we minimize the sum of squares of errors: $\min E = \min (\Sigma^m e_l^2)$. The vector of parameters $(k_{11},\ldots, k_{np}, t_{11},\ldots,t_{np})$ should be adjusted according to the gradient vector$(dE^2/dk_{11},\ldots,dE^2/dk_{np},dE^2/dt_{11},\ldots,dE^2/dt_{np})$: $k_{ij} = k_{ij} - c \times dE^2/dk_{ij}$, $t_{ij} = t_{ij} - c \times dE^2/dt_{ij}$, where c is some constant. Iteratively repeating this procedure, we get an algorithm for adjustment of the parameters of the fuzzy system.

$$dE^2/dk_{ij} = d(\Sigma_l^m e_l^2)/dk_{ij} =$$

$$= d(\Sigma_l^m(y_l - \Sigma_j(r_j \times max_j(min_i(h_{ij}(x_{il}))) \times$$

$$\times (1 + \mathrm{sgn}(min_i(h_{ij}(x_i)) - max_j(min_i(h_{ij}(x_{il})))))))^2/dk_{ij} =$$

$$= d(\Sigma_l^m y_l - \Sigma_j(r_j max_j(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1))(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)))) \times (1 + \mathrm{sgn}(min_i(h_{ij}(x_i)) - max_j(min_i(h_{ij}(x_{il})))))))^2/dk_{ij} =$$

$$= (-2) \times (\Sigma_l^m(y_l - \Sigma_j(r_j \times max_j(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1))(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) \times (\Sigma_j(r_j \times ((1 + \mathrm{sgn}(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 -$$

$\mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) - max_j(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1))(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)))\times(1 + \mathrm{sgn}(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) - ((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4))\times((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)/4 \times x_{il} - (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il})/4 \times x_{il})))) = (-2)\times(\Sigma_l^m(e_l \times (\Sigma_j(r_j\times(1 + \mathrm{sgn}(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) - max_j(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1))(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)))\times(\Sigma_j(r_j\times(1 + \mathrm{sgn}(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) - ((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4))\times((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)/4 \times x_{il} - (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il})/4 \times x_{il})))) = (-2)\times\Sigma_l^m(e_l \times \Sigma_j(r_j\times I_{ij}(x_{il}) \times x_{il})),$

$$\text{where } I_{ij}(x_{il}) = \begin{cases} 1, \text{ if } x_{il} \text{ belongs to the left side of the triangle describing the fuzzy set } max_j(min_i(h_{ij}(x_{il}))) \\ -1, \text{ if } x_{il} \text{ belongs to the left side of the triangle describing the fuzzy set } max_j(min_i(h_{ij}(x_{il}))) \\ 0, \text{ otherwise.} \end{cases}$$

$dE^2/dt_{ij} = d(\Sigma_l^m e_l^2)/dt_{ij} =$

$= d(\Sigma_l^m(y_l - \Sigma_j(r_j\times max_j(min_i(h_{ij}(x_{il}))))))^2/dt_{ij} =$

$= d(\Sigma_l^m(y_l - \Sigma_j(r_j\times max_j(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1))(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)))))))^2/dt_{ij} =$

$= (-2)\cdot(\Sigma_l^m(y_l - \Sigma_j(r_j\times max_j(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1))(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) - max_j(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1))(k_{ij}x_{il} - t_{ij})/4 + (1 + \mathrm{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \mathrm{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)))\cdot(1 + \mathrm{sgn}(min_i((1 + \mathrm{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \mathrm{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1$

$+ \operatorname{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \operatorname{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) - ((1 + \operatorname{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \operatorname{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \operatorname{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \operatorname{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) \cdot (-(1 + \operatorname{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \operatorname{sgn}(k_{ij}x_{il} - t_{ij} - 1)/4 + (1 + \operatorname{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \operatorname{sgn}(1 + t_{ij} - k_{ij}x_{il})/4))) = (-2) \times (\Sigma_l^m(e_l \cdot (\Sigma_j(r_j \times (1 + \operatorname{sgn}(min_i((1 + \operatorname{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \operatorname{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \operatorname{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \operatorname{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) - max_j(min_i((1 + \operatorname{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \operatorname{sgn}(k_{ij}x_{il} - t_{ij} - 1))(k_{ij}x_{il} - t_{ij})/4 + (1 + \operatorname{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \operatorname{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4))) \cdot (1 + \operatorname{sgn}(min_i((1 + \operatorname{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \operatorname{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \operatorname{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \operatorname{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) - ((1 + \operatorname{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \operatorname{sgn}(k_{ij}x_{il} - t_{ij} - 1)(k_{ij}x_{il} - t_{ij})/4 + (1 + \operatorname{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \operatorname{sgn}(1 + t_{ij} - k_{ij}x_{il}))(2 + t_{ij} - k_{ij}x_{il})/4)) \cdot (-(1 + \operatorname{sgn}(k_{ij}x_{il} - t_{ij}))(1 - \operatorname{sgn}(k_{ij}x_{il} - t_{ij} - 1)/4 + (1 + \operatorname{sgn}(2 + t_{ij} - k_{ij}x_{il}))(1 - \operatorname{sgn}(1 + t_{ij} - k_{ij}x_{il})/4))) = (-2) \cdot \Sigma_l^m(e_l \cdot \Sigma_j(r_j \cdot I_{ij}(x_{il}))).$

The gradient vector is evaluated for each iteration. Thus, for iteration $p+1$ $k_{ij(p+1)} = k_{ijp} - c \times dE^2/dk_{ijp}$, $t_{ij(p+1)} = t_{ijp} - c \times dE^2/dt_{ijp}$. This process continues until the sum of the absolute values of the differences $\Sigma_i\Sigma_j(|k_{ij(p+1)} - k_{ijp}|/k_{ijp} + |t_{ij(p+1)} - t_{ijp}|/t_{ijp})$ would be less than a fixed error tolerance ET.

We can adjust the parameter r_j of the consequent during this iteration. This adjustment is correct only for the maximum-membership aggregation.

$$dE^2/dt_{ij} = d(\Sigma_l^m e_l^2)/dr_j =$$
$$= d(\Sigma_l^m(y_l - \Sigma_j(r_j \times max_j(min_i(h_{ij}(x_{il})))))^2/dr_j =$$
$$= (-2) \cdot \Sigma_l^m(y_l - \Sigma_j(r_j \times max_j(min_i(h_{ij}(x_{il})))) \cdot max_j(min_i(h_{ij}(x_{il})))) =$$
$$= (-2) \cdot \Sigma_l^m e_l \cdot max_j(min_i(h_{ij}(x_{il}))),$$

The iteration step is computed as $r_{j(p+1)} = r_{jp} - c \times dE^2/dr_{jp}$.

It is extremely important to emphasize that at this stage our main concern is not so much convergence of the process but the definitions of fuzzy sets describing antecedents of a fuzzy system. For example, applying the above algorithm to our example with input-output triples $<12;0.4,25>$, $<8;0.7;40>$, $<5,0.9.68>$, $<17;0.2;15>$ gives the fuzzy sets for the angle of the road and friction as in fig. 7.7 in appendix.

One point should be made clear. After the iteration we compute the modes of each fuzzy set (the value of the argument at which membership function is equal 1). The formula is

$M_{ij} = (1 + t_{ij})/k_{ij}$.

The difference between this mode and the initial mode M_{ij}^0 given by experts should be less than the difference between this mode and the mode of any other linguistic set. Otherwise, we can't claim that linguistic variables (positive medium, negative small, etc) are the same that were used in the initial system of experts. In this case, rules with such fuzzy sets should be modified. For example, if we discover that the computed mode of fuzzy set 'positive small' of variable x_i is closer to the mode of fuzzy 'positive medium' we should modify rules: substitute 'positive medium for 'positive big'.

The resulting fuzzy system consists of rules in which antecedents of rules are closer to those used in human reasoning. The above procedure provides an exact definition of what people do mean when they use expressions such as "if variable x is big". Such definitions are ones that produce minimal errors in a fuzzy system. The next step is to find parameters defining consequents.

7.4. Estimation of Parameters of Consequents.

Maximum-membership aggregation ignores much of the information in original data [Kosko, 1993]. This scheme chooses the element r_{max} with maximum membership in the fuzzy sets defining consequents: $m(r_{max}) = max_j(m(r_j))$. The resulting parameter r_{max} tends not to have unique value. Another problem is that this type of aggregation loses most of the information in the system. The alternative is to use a fuzzy centroid scheme. This scheme computes output as

$$(*) \qquad d = \Sigma_j(r_j \cdot m(r_j))/\Sigma_j m(r_j).$$

This output is unique and uses all the information in the system. We want to find optimal parameters of consequents in a fuzzy system. These parameters can be identified as r_j.

Our problem is to find such r_j that generate the least error $e = (y - d)$, where y is known output. We can represent the equation (*) as $d = \Sigma_j r_j \cdot (m(r_j)/\Sigma_i m(r_j))$. In this equation $m(r_j)$ is actually $min_i(h_{ij}(x_{il}))$ and doesn't depend on r_j itself. Therefore, we consider them as parameters and denote $m(r_j)/\Sigma_j m(r_j)$ by u_j. The equation (*) is now

$$(**) \qquad d = \Sigma_j r_j \cdot u_j.$$

We want to minimize $e^2 = (y - d)^2 = (y - \Sigma_j r_j \cdot u_j)^2$.

This problem is a standard one for statistics. It can be solved using the **least-mean-square (LMS)** algorithm. This algorithm computes the value of the parameter r_j at an iteration step k+1:

$$r_{j(k+1)} = r_{jk} + 2c \cdot e_k \cdot u_{jk}.$$

7.4. Competitive Neural Networks and Fuzzy Systems.

Competitive neural networks function in the same way as standard neural networks, but the process of learning is different. In competitive neural networks, only one output unit is active at the end of each learning cycle. Afterwards, weights on the input lines that lead to this single output unit are adjusted. These steps (finding the most active unit and adjusting the weights) correspond precisely to the adjustment of parameters in our algorithm for adjustment parameters in fuzzy systems. The reason is that the most active unit is the unit which has maximum activation level, which corresponds exactly to fuzzy systems with maximum aggregation scheme.

Thus, fuzzy systems in our algorithm simulate competitive neural networks. A common application of competitive neural networks is learning of unknown patterns. The corresponding fuzzy system provides rules which are, unlike the neural network, transparent.

7.5. Formalization of Neural Networks.

If neural networks can be simulated by fuzzy systems then it may be possible to apply the method of formalization of fuzzy systems to neural networks. In neural nets, parameters (weights) define the working of a particular network. For each input $<x_1,x_2,\ldots,x_n>$ there is a corresponding output y: $g(<x_1,x_2,\ldots,x_n>) = y$. These functions form objects in a category. Morphisms in these category are functions y--->y' such that the triangle in fig 7.6 in appendix commutes. The output of a neural network depends on weights. These weights define the performance of the input-output function. We can represent them as functors: $F(W)$----->$(g<x_1,x_2,\ldots,x_n>$ ---> y). Adjustment of weights using backpropagation changes weights from W to W'. This corresponds to a diagram for morphisms in fig 7.8 in appendix.

APPENDIX

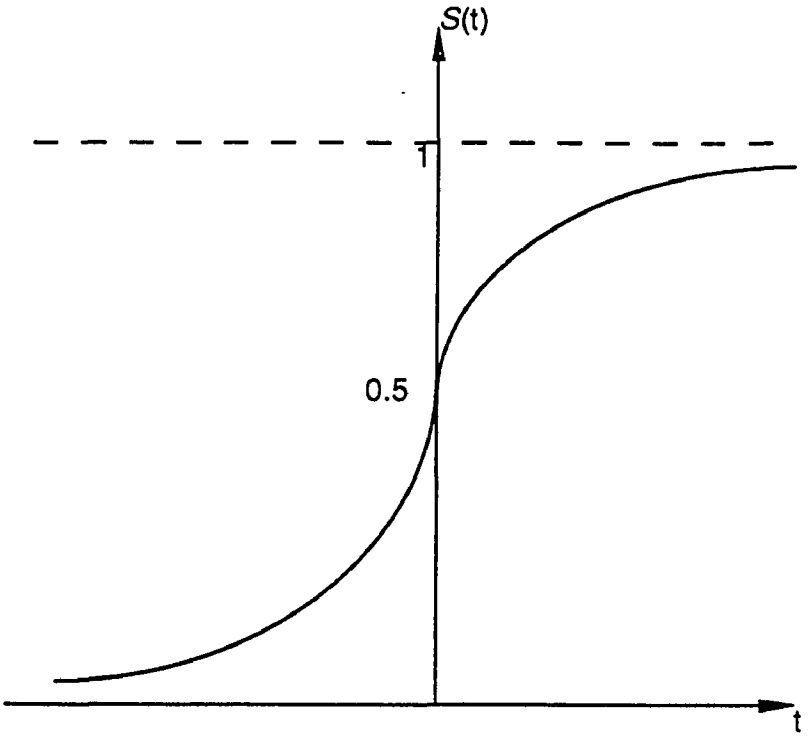

Figure 6.1. The sigmoid function

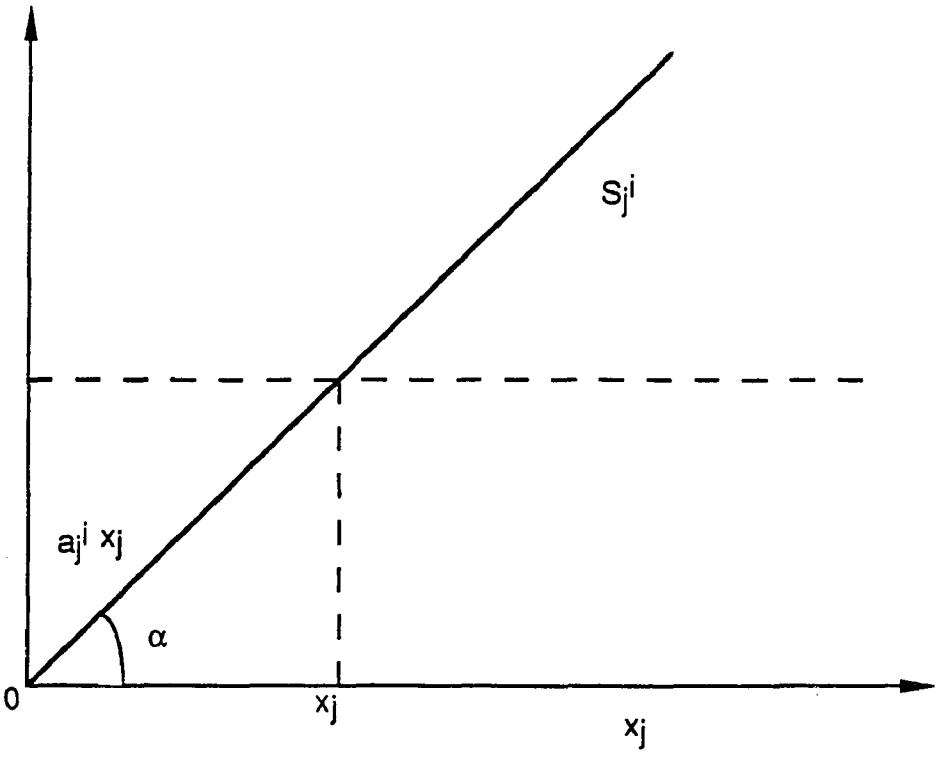

Figure 6.2. Fuzzy set S_j^i simulated by $a_j^i x_j$

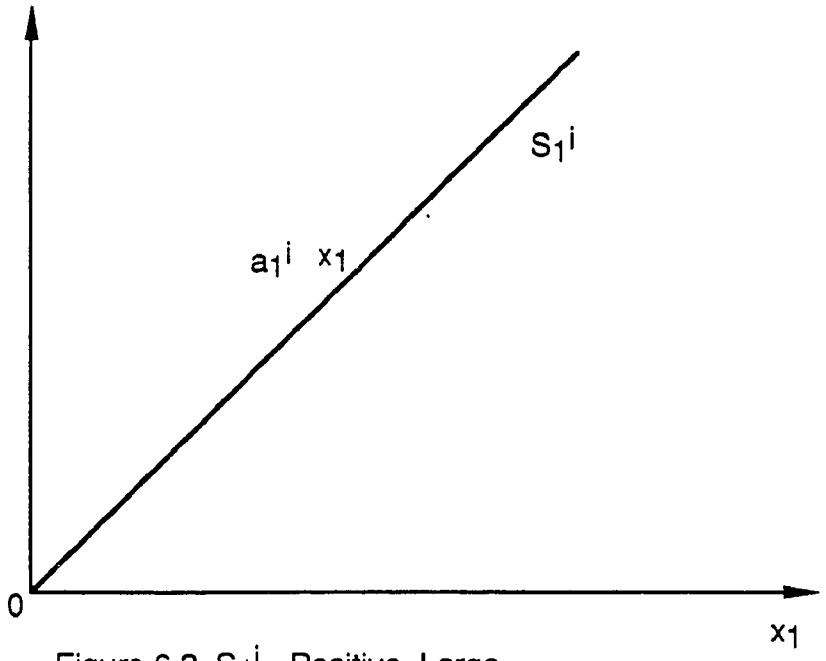

Figure 6.3. S_1^i - Positive Large

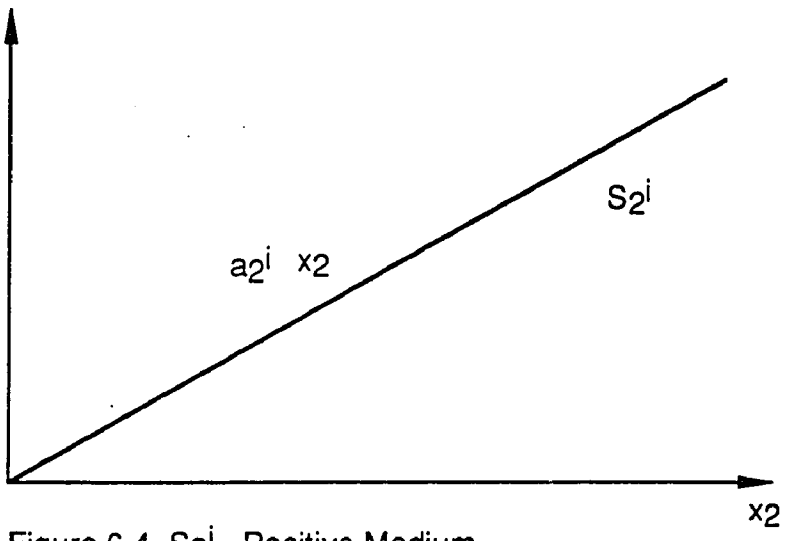

Figure 6.4. S_2^i - Positive Medium

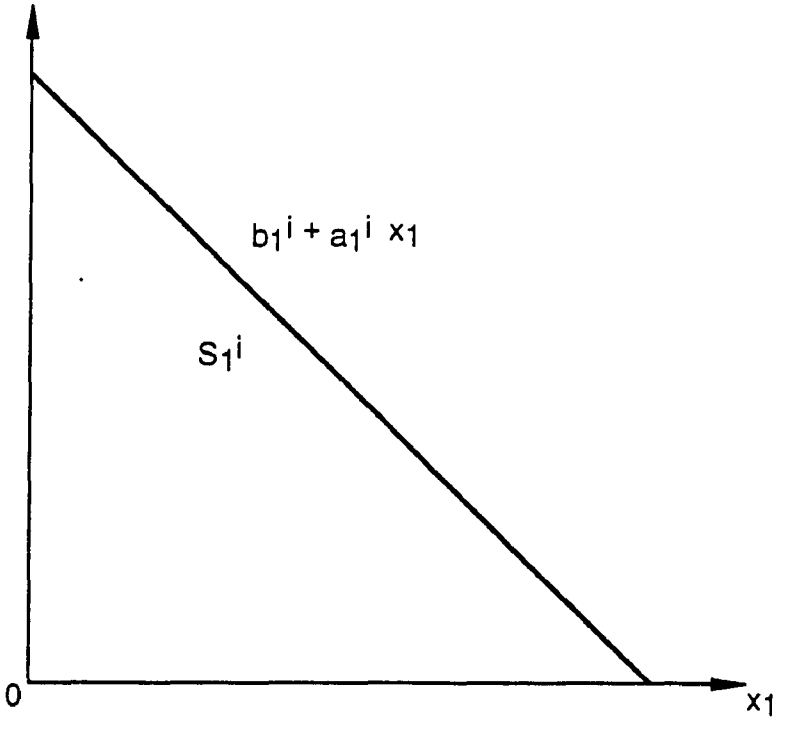

Figure 6.5. Fuzzy set S_1^i - Negative medium

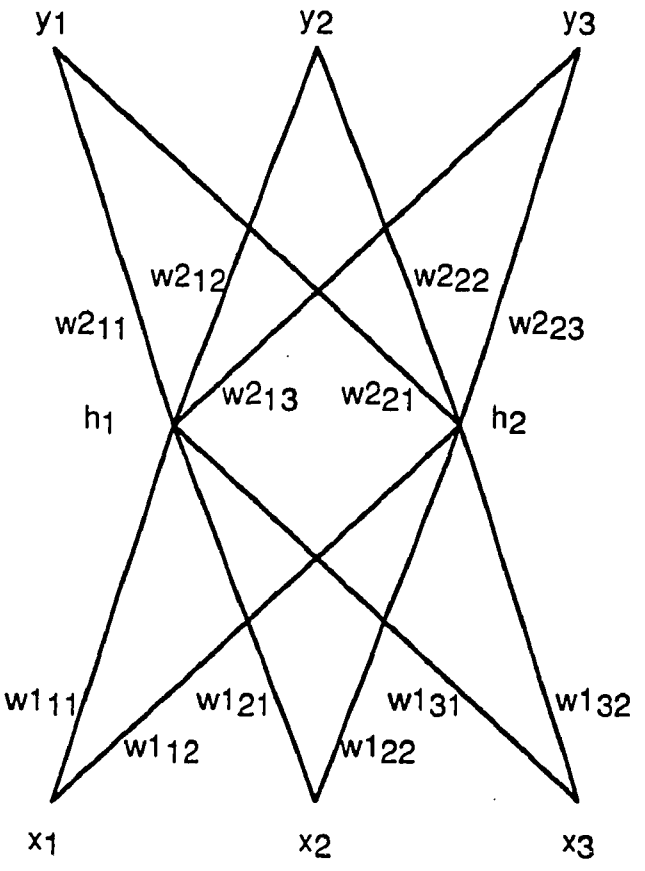

figure 6.6. Neural Network 3 - 2 - 3

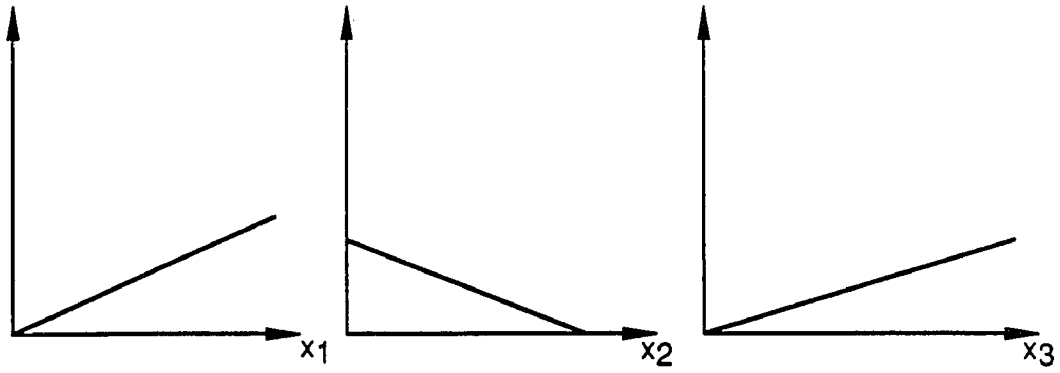

Figure 6.7. Fuzzy sets corresponding to Rule 1.

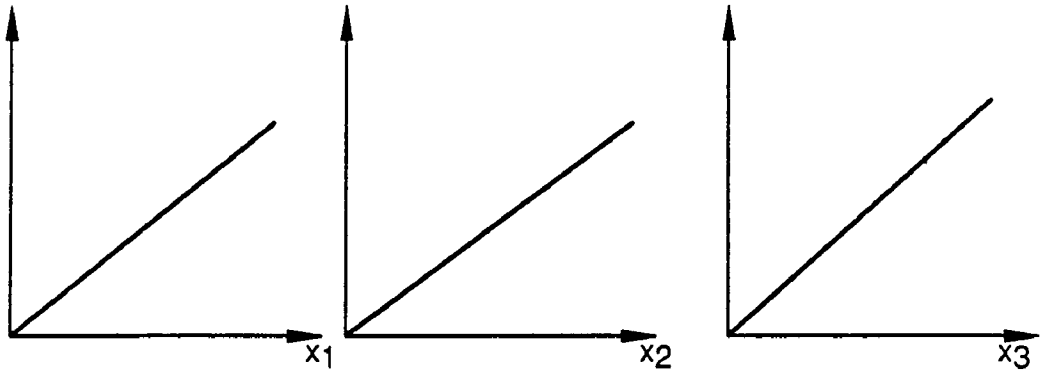

Figure 6.8. Fuzzy sets corresponding to Rule 2.

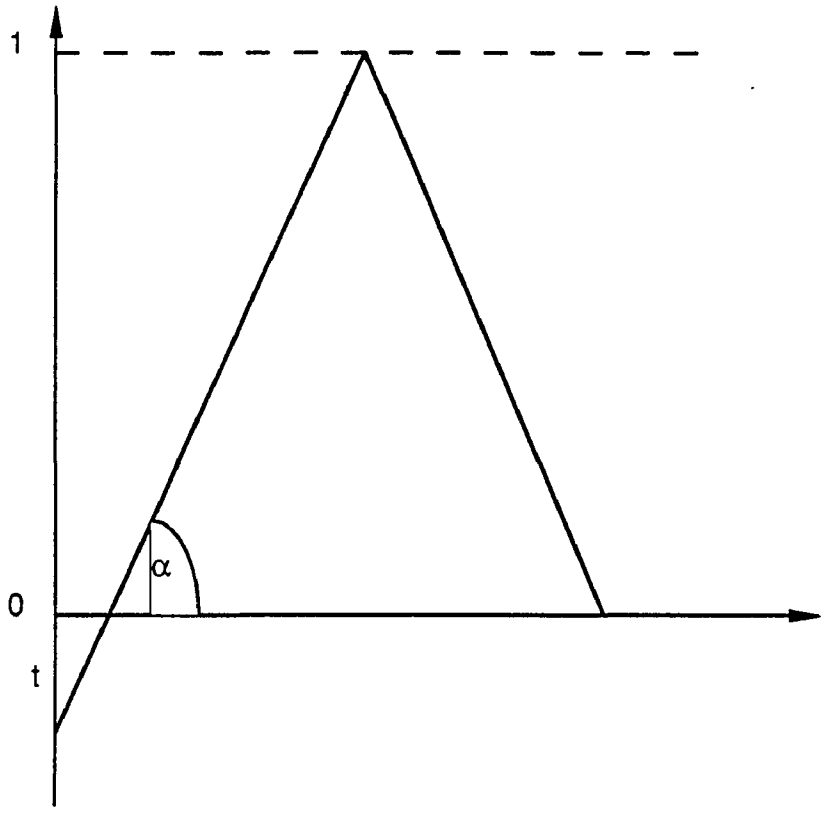

Figure 7.1. Triangle basis spline.

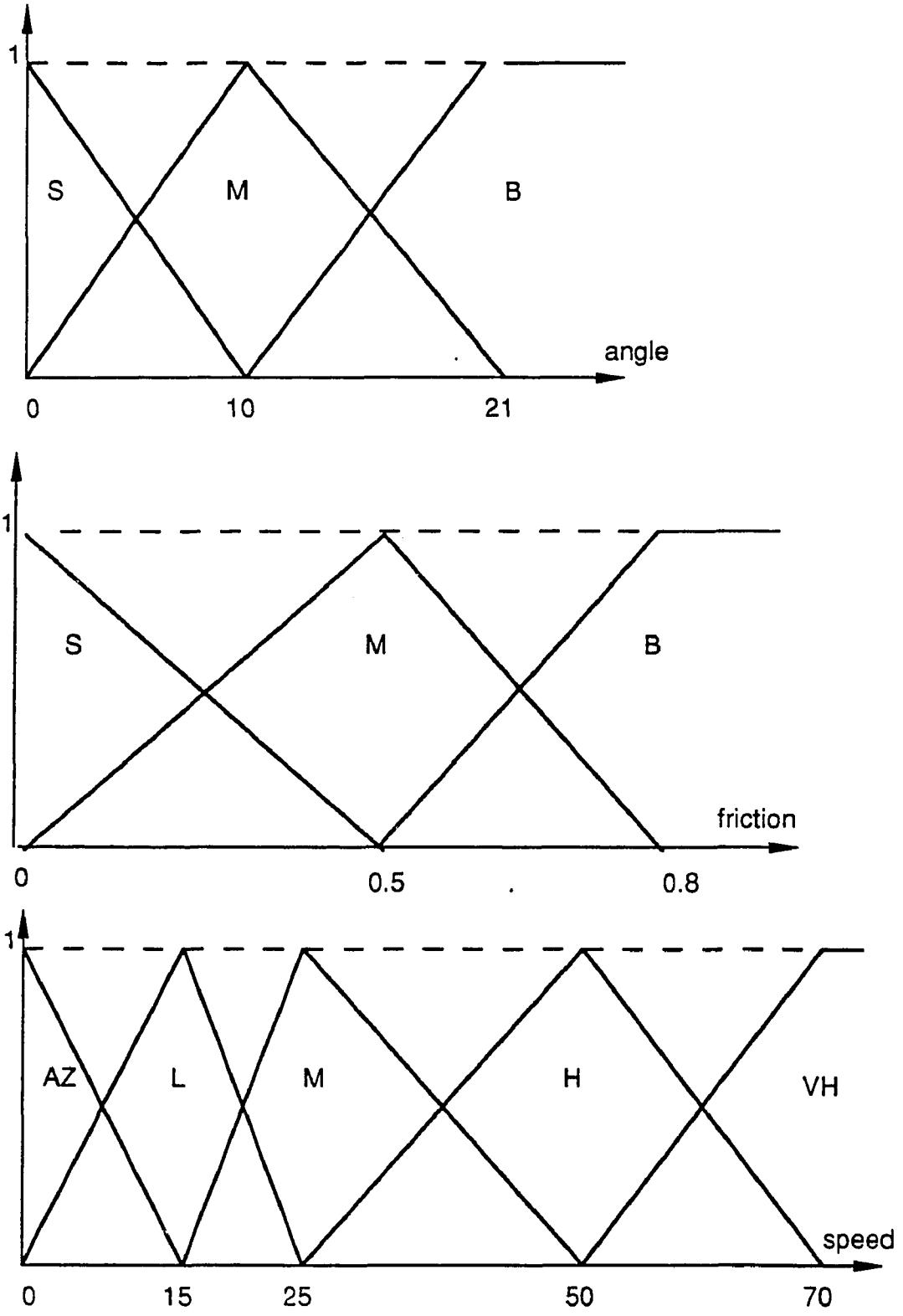

Figure 7.2. Fuzzy sets corresponding to angle of the road, friction, speed

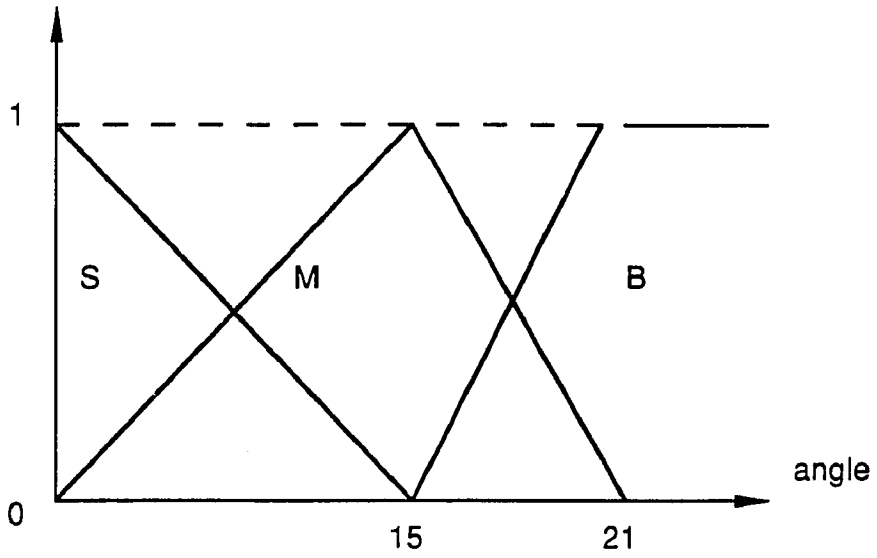

Fig. 7.3. Modified fuzzy set corresponding to angle of the road.

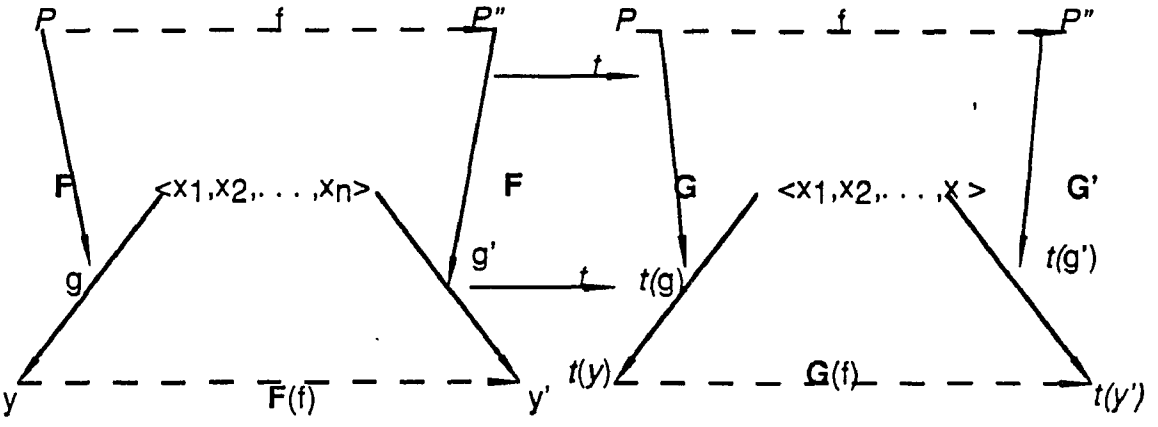

Figure 7.6. Functors in the category of fuzzy systems

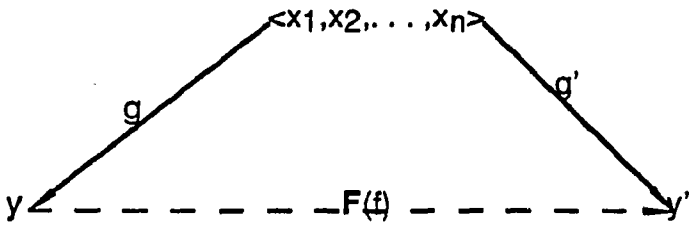

Figure 7.4

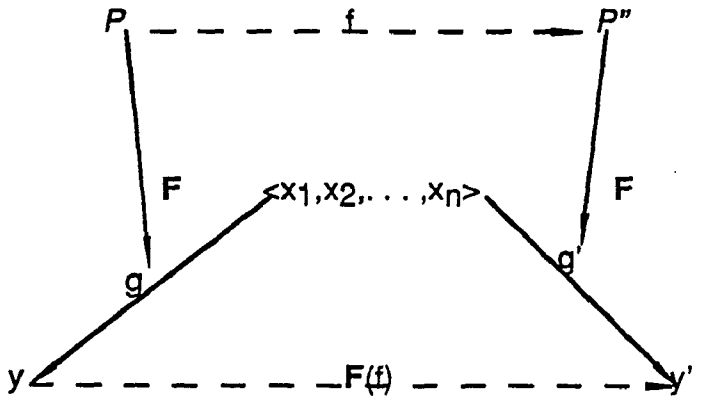

Figure 7.5

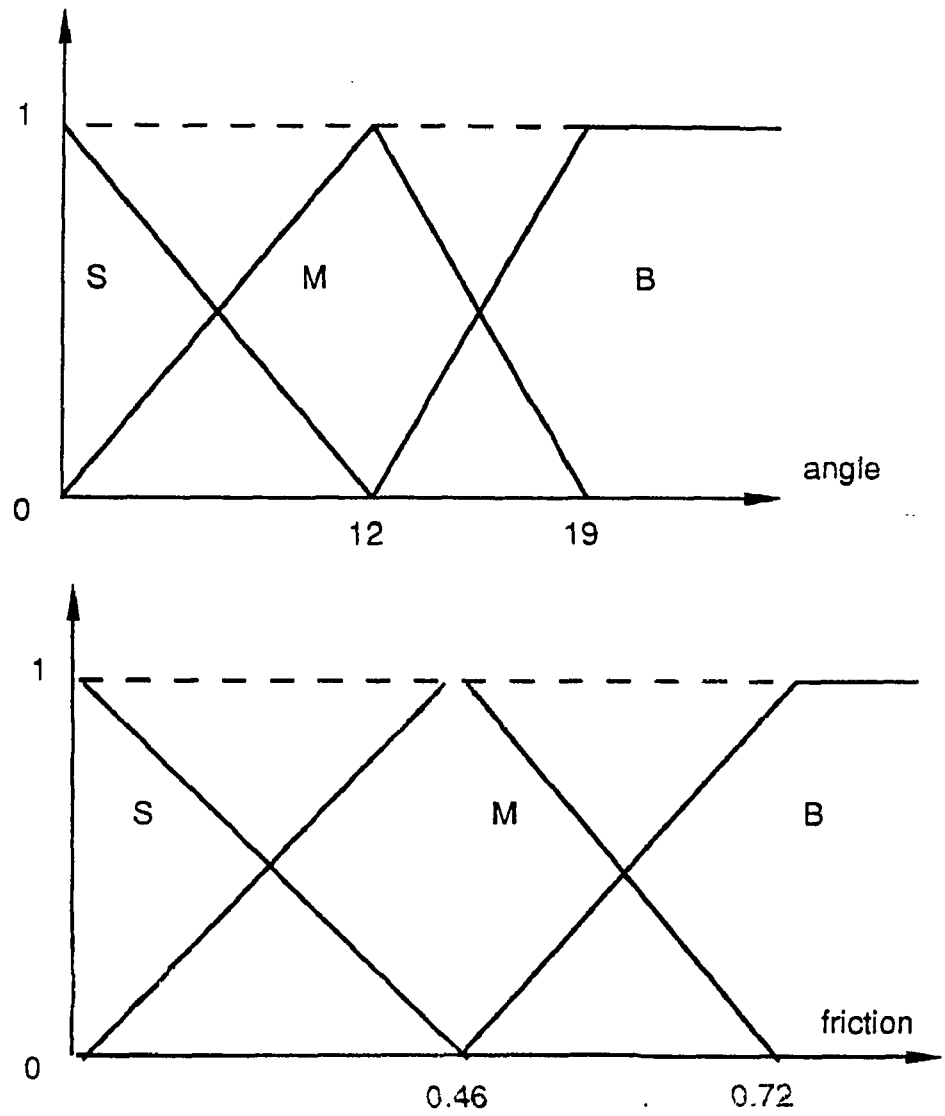

Figure 7.7. Fuzzy sets for angle of the road and friction, modified by the algorithm.

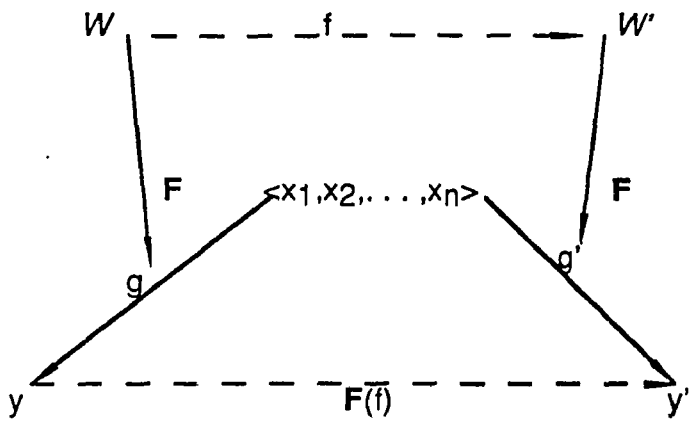

figure 7.8. Morphisms in the category of neural networks.

Bibliography.

Abidi, M.A. and Gonsales, R.C., "Data fusion and machine intelligence", Academic Press, Boston, 1992

Alsina C., Trillas, E., and Valverde, L., "On some logical connectives for fuzzy set theory", Journal of Mathematical Analysis and Applications, 93 (1983), 15-26

"Approximate reasoning in intelligent systems", Oxford, Pergamon Press, 1987

Asai, K., Sugeno, M., Terano, T., "Applied fuzzy systems", Academic Press, New York, 1994

Bortolan, G., "Neural networks for processing of fuzzy sets", ICANN'94, 181-184

Buckley, J.J., Hayashi, Y., Ghozala, E., "On equivalence of neural nets and fuzzy expert systems", Proceedings of IEEE International Conference on neural networks, June 7-12, 1992, volume 2, 691-695

Buckley, J.J., Hayashi, Y., "Can fuzzy neural networks approximate continuous fuzzy systems", Fuzzy sets and systems, 61, 43-53, 1993

Brown, R.G., Smoothing J., "Forecasting and prediction of discrete time series", Prentice Hall, Englewood Cliffs, NJ, 1963

Ding, H., Gupta, M. "Foundations of fuzzy neutral computation"
SCFL, 165-169 - in Amizadeh, F., Jamshiadi, M. (Eds.), "Soft computing - fuzzy logic, neural networks and distributed artificial intelligence", Prentice Hall, Englewood Cliffs, NJ, 1994

Dubois, D., Prade, H., "A review of fuzzy sets aggregation connectives", Information Sciences, 36(1985), 85-21

Dyckoff H., Pedrycz, W., "Generalized means as model of compensative connectives", Fuzzy sets and systems, 14(1984), 143-154

Elcan, C., "The paradoxical success of fuzzy logic", IEEE Expert, 1995

Feigenbaum, E.A., Feldman, J., "Computers and thought",
McGrow-Hill Book Company, New York, 1963

Fodor, J.C., Yager, R.R. and Rybalov A., "Structure of uni-norms," Technical Report #MII-1501, Machine Intelligence Institute, Iona College, New Rochelle, NY, 1985

"Fuzzy logic for management and uncertainty", Wiley, New York, 1992

"Fuzzy expert systems", CRC Press, Boca Raton, FL, 1992

"Fuzzy relations equations and their applications to knowledge engineering", Kluwer Academic, Dordrecht; Boston 1989.

Hart, W.L., Analytic Geometry, D.C. Heath & Co.: Boston, 1957.

Hasegawa, T., Howikawa, S.,Furubashi, T., Uchikawa, Y., "On design of adaptive fuzzy controller using fuzzy neural network and a description of its dynamic behavior", Fuzzy sets and systems, 71, 1994

Harris, C.J.,"Intelligent control: aspects of fuzzy logic and neural networks", World Scientific, Singapore; River Edge, NJ, 1993

Hertz, D.H., Hu, Q., Fuzzy logic controlled neural network learning", Information science, 2(1994), 15-33

E.P. Klement, "Characterization of fuzzy measures constructed by means of triangular norms," Journal of Math. Analysis and Application, 86(1992), 345-358

Klir, G.J., Folger, T.A., "Fuzzy sets, uncertainty and information", Prentice Hall, Englewood Cliffs, NJ, 1988

Kosko, B., "Fuzzy systems as universal approximation", Proceedings of IEEE International Conference on fuzzy systems, March 8-12, 1992, San Diego, 1153-1162

Kosko, B., "Neural networks and fuzzy systems: a dynamical approach to machine intelligence", Prentice Hall, Englewood Cliffs, NJ, 1992

Kosko, B., "Fuzzy thinking: the new science of fuzzy logic", Hyperion, New York, 1993

Lee, C.C., "Fuzzy logic in control systems: fuzzy logic controller", IEEE Transactions on Systems, Man and Cybernetics, 20, 404-418, 1990

Lin, C.T., "Neural fuzzy control systems with structure and parameter learning", World Scientific, Singapore, Piever Edge, NJ, 1994

Mamdani, E.H., Assilian, S., "An experiment in linguistic synthesis with a fuzzy logic controller", International Journal of Man-Machine Studies, 7(1975), 1-13

McNeil, D., "Fuzzy logic", New York, Simon and Shuster, 1993

Negoita, C.V., "Expert systems and fuzzy systems", Benjamin/Cummings Publishing Company, Menlo Park, CA, 1985

Negoita, C.V., "Neural networks and fuzzy systems", Kybernetes, **23**(3), 1994, 7-9

"Non-standard logic for automated reasoning", Academic Press, San Diego, CA, 1988

Pedrycz, W., "Fuzzy logic and fuzzy systems", Wiley, New York, 1994

Pedrycz, W., "Fuzzy neural networks: concepts, models, optimizations", FO, 406-420, in Delgado, M., Kacprzyk, J., Veregay, J.L., Villa M.A. (Eds.) "Fuzzy optimization ", Physical, Heidelberg, 1994

Sugeno, M., Kang G.T., "Structure identification of fuzzy models", Fuzzy Sets and Systems, 28(1988), 15-33

Takagi, T., Sugeno, M., " Fuzzy identification of systems and its application to modeling and control", IEEE Transaction on Systems, Man and Cybernetics, 15(1985), 116-132

Terano, T., "Fuzzy systems theory and its applications", Academic Press, Boston, 1992

Weber,S., "A general concept of fuzzy connectives, negations and implications, based

on t-norms", Fuzzy Sets and Systems, 11(1983), 115-134

Yager, R.R., "New direction in multicensor fusion", Proceedings of SPIE Conference on Advances in Intelligent Robotic Systems, Cambridge, MA, 1988, 405-415

Yager, R.R., "Aggregation operators and fuzzy systems modeling", Fuzzy Sets and Systems, 67(1994), 129-146

Yager, R.R., "On inference structure for fuzzy system modeling", Proceedings of the Third International Conference on Fuzzy Systems, Orlando, 1994, 1252-1256

Yager, R.R., "On mean type aggregation", IEEE Transactions on Systems, Man and Cybernetics (to appear)

Yager R. R., "A general approach to the fusion of imprecise information", Technical report # MI-1512, Machine Intelligence Institute, Iona College, New Rochelle, NY, 1995

Yager, R.R., Filev, D.P., "Essentials of Fuzzy modeling and control", John Wiley, NY, 1994

Yager, R.R., "MAM and MOM operators for aggregation", Information Sciences, 69(1993), 259-273

Yager, R.R., Fodor J.C., Rybalov, A., "Structure of uni-norms" (to appear in Information Science)

Yager, R.R., Rybalov, A., "Uni-norm aggregation operators" (to appear in Fuzzy Sets and Systems)

Yager, R.R., Rybalov, A., "Full reinforcement operators in aggregation techniques", (to appear in IEEE Transactions on Systems, Man and Cybernetics)

Yager, R.R., Rybalov, A., "Non commutative self identity operators" (to appear in Fuzzy Sets and Systems)

Zadeh, L.A., "A theory of approximate reasoning", in Hayes, J., Michie, D., Mikulich, L.I. (Eds.), Machine Intelligence, 9(1979), Halstead Press, New York, 149-194

Zadeh, L.A., " A computational approach to fuzzy quantifiers in natural languages", Computing and Mathematics with applications, 9(1983), 149-184