Induced Ordered Weighted Averaging Operators

Ronald R. Yager, *Fellow, IEEE,* and Dimitar P. Filev, *Senior Member, IEEE*

***Abstract*—We briefly describe the Ordered Weighted Averaging (OWA) operator and discuss a methodology for learning the associated weighting vector from observational data. We then introduce a more general type of OWA operator called the Induced Ordered Weighted Averaging (IOWA) Operator. These operators take as their argument pairs, called OWA pairs, in which one component is used to induce an ordering over the second components which are then aggregated. A number of different aggregation situations have been shown to be representable in this framework. We then show how this tool can be used to represent different types of aggregation models.**

***Index Terms*—Aggregation, fuzzy, ordered weighted averaging.**

I. Introduction

The **O**rdered **W**eighted **A**veraging (OWA) operators [1], [2] provide a parameterized family of mean type aggregation operators. An important feature of these operators is the reordering step, which makes this a nonlinear operator. During this step the arguments are ordered by their value. In [3] the authors suggested an alternative approach to ordering the arguments based upon using a function of the arguments rather then the arguments themselves. Motivated by this idea we consider here a general class of OWA operators in which the ordering of the arguments is induced by another variable called the order inducing variable. As we shall see these induced OWA operators, which essentially aggregate objects which are pairs, provide a very general family of aggregation operators. Particularly note worthy are their ability to provide for aggregations in environments which mix ordinal (i.e. linguistic) and numeric variables. This facility for mixing these different data types can be seen to be very useful in helping accomplish the agenda of computing with words specified by Zadeh [4].

II. The OWA Operator

In the following, we provide a definition of the OWA operators as introduced by Yager [1].

Definition: An OWA operator of dimension n is a mapping

$$F: R^n \rightarrow R^n$$

that has an associated weighting vector W of dimension n having the properties

$$w_j \in [0,1] \tag{1}$$

$$\sum_{j=1}^{n} w_j = 1 \tag{2}$$

and such that

$$F(a_1, \cdots, a_n) = \sum_{j=1}^{n} w_j b_j$$

where b_j is the jth largest of the a_i.

Central to this operator is the reordering of the arguments, based upon their values. That is, the weights rather than being associated with a specific argument, as in the case of the usual weighted average, are associated with a particular position in the ordering. We note this reordering introduces a nonlinearity into an otherwise linear process.

If B is a vector corresponding to the ordered arguments, we shall call this the ordered argument vector, and W^T is the transpose of the weighing vector then we can express the OWA aggregation in vector notation as

$$F_w(a_1, \cdots, a_n) = W^T B.$$

As noted in the literature [5], the OWA operator provides a very rich family of aggregation operators parameterized by the weighting vector. Among the operators included in this family are the average, max, min and median of the arguments. It was also shown in [1] that the OWA operator is a mean operator as it satisfies the following conditions.

1) **Monotonicity:** $F(a_1, \cdots, a_n) \geq F(\hat{a}_1, \cdots, \hat{a}_n)$ if $a_i \geq \hat{a}_i$ for i.
2) **Commutativity:** the initial indexing of the arguments doesn't matter.
3) **Bounded:** $\min_j[a_j] \leq F(a_1, \cdots, a_n) \leq \max_j[a_j]$.

It can also be shown that this operator is idempotent, if $a_j = a$ for all j, then $F(a_1, \cdots, a_n) = a$.

One basic application using these OWA operators is that of providing a fusion of the arguments where the selection of the weights can be used to model some aggregation imperative. Consider a scenario in which we are trying to determine the number of enemy airplanes that are coming to attack us. Assume we have n equally reliable sources of information, sensors, each providing an estimate of the number of enemy planes. Here we must combine the individual sensor estimates to obtain a single fused estimate. A natural approach would be to take the average, however, because in this situation under estimating the power of the enemy force would be more costly than over estimating it, we may want to give more weight to

Manuscript received March 1, 1998; revised August 30, 1998. This paper was recommended by Associate Editor A. Kandel.

R. R. Yager is with the Machine Intelligence Institute, Iona College, New Rochelle, NY 10801 USA (e-mail: ryager@iona.edu).

D. P. Filev is with the Ford Motor Company, Detroit, MI 48239 USA.

Publisher Item Identifier S 1083-4419(99)02300-6.

the higher valued estimates, this of course can be accomplished using the OWA operator.

In [1], Yager introduced two characterizing measures associated with the weighting vector W of an OWA operator. The first of these which we shall call the *alpha* value of W, denoted as $\alpha(W)$, is defined as

$$\alpha(W) = \frac{1}{n-1} \sum_{j=1}^{n} w_j(n-j).$$

This measure which takes it value in the unit interval can be seen to reflect the degree to which W gives preference to the bigger or smaller arguments in the aggregation. It can be shown that when W is the max aggregation $\alpha(W) = 1$, while when W is the min the α value is zero and when W is the average, α is 0.5. While the semantics associated with α may depend upon the applications, generally $\alpha(W)$ can be seen to measure the degree of *maxness* of the aggregation, how much it is like a pure max operator.

The second measure introduced by Yager [1]

$$H(W) = -\sum_{j=1}^{n} w_j \ln(w_i)$$

is called the measure of dispersion (or entropy) and was shown to measure the degree to which W takes into account all information in the aggregation.

For example, consider the three weighting vectors

$$W-1 = \begin{bmatrix} 0.2\\0.2\\0.2\\0.2\\0.2 \end{bmatrix} \quad W-2 = \begin{bmatrix} 0.5\\0\\0\\0\\0.5 \end{bmatrix} \quad W-3 = \begin{bmatrix} 0\\0\\1\\0\\0 \end{bmatrix}.$$

All three of these can be shown to have the same measure alpha value

$$\alpha(W-1) = \alpha(W-2) = \alpha(W-3) = 0.5$$

however they have different measures of dispersion

$$H(W-1) = -\sum_{j=1}^{5} \tfrac{1}{5} \ln(\tfrac{1}{5}) = -\ln(\tfrac{1}{5}) = \ln(5)$$
$$H(W-2) = -\ln(\tfrac{1}{2}) = \ln(2)$$
$$H(W-3) = 0$$

We see that the more uniform the weights the higher the dispersion.

III. Learning the OWA Weights

The actual type of aggregation performed by an OWA operator depends upon the form of the weighting vector W. Using $w_j = (1/n)$ gives us the simple average while having $w_1 = 1$ and $w_j = 0$ for $j \neq 1$ gives us the max of the arguments and if $w_n = 1$ and $w_j = 0$ for $j \neq n$ we obtain the min of the arguments. The process of determining the weighting vector is important and is discussed at considerable length in [6]. One method for obtaining the weighting vector is to associate the OWA aggregation with a linguistic quantifier represented as a fuzzy subset Q on the unit interval [1]. In this approach, the weights associated with a aggregation of degree n are obtained as

$$w_j = Q\left(\frac{j}{n}\right) - Q\left(\frac{j-1}{n}\right) \quad \text{for} \quad j = 1, \cdots, n.$$

Another approach, suggested by O'Hagan [7], for obtaining the weights makes use of the two measures previously introduced and is based upon the solution of a mathematical programming problem. In this approach the user supplies a value for $\alpha \in [0,1]$ indicating the degree of "maxness" described in the aggregation. The weights are then obtained by solving the following programming problem:

$$\textbf{Max}: -\sum_{j=1}^{n} w_j \ln(w_i).$$

Subject to

1. $\alpha = \frac{1}{n-1} \sum_{j=1}^{n} w_j(n-j)$

2. $\sum_{j=1}^{n} w_j = 1$

3. $0 \le w_j \le 1 \quad \text{for} \quad j = 1, \cdots, n.$

O'Hagan [7] called the weights obtained for a given α the ME-OWA (maximal entropy) weights.

In [8] and [9] Filev and Yager introduced a technique for learning the weighting vector associated with an OWA aggregation from observational data. This technique is very much in the spirit of the types of learning algorithms used in neural networks [10], [11] and is also based upon the gradient descent method. In the following we describe this algorithm for learning the n weights of the OWA weighting vector from observations.

Assume we have a collection of m samples each comprised of an n-tuple of values $(a_{k1}, a_{k2}, \cdots, a_{kn})$, called the arguments, and an associated single value called the aggregated value, which we shall denote as d_k. Our goal here is to construct, using the OWA aggregation, a model that adequately represents this process. Essentially the problem becomes that of learning a weighting vector W, that models the process of aggregation over the data set. We shall learn the weighting vector W from the given data.

The problem of learning the weights can be simplified by taking advantage of the linearity of the OWA aggregation with respect to the ordered arguments. We denote the reordered objects of the kth sample by $b_{k1}, b_{k2}, \cdots, b_{kn}$ where b_{kj} is the jth largest element of the argument collection $\langle a_{k1}, a_{k2}, \cdots, a_{kn}\rangle$. Using these ordered arguments the problem of modeling the aggregation process is to find the vector of OWA weights $W^T = [w_1\ w_2 \cdots w_n]$ such that

$$b_{k1}w_1 + b_{k2}w_2 + \cdots + b_{kn}w_n = d_k$$

for every $k = 1$ to m.

We shall relax this formulation by looking for a vector of OWA weights W that approximates the aggregation operator by minimizing the instantaneous errors e_k

$$e_k = \tfrac{1}{2}\left(b_{k1}w_1 + b_{k2}w_2 + \cdots + b_{kn}w_n - d_k\right)^2$$

with respect to weights w_i. This learning problem is, in fact, a constrained optimization problem, since the OWA weights w_i have to satisfy the following two properties:

1. $\sum_{i=1}^{n} w_i = 1$;
2. $w_i \in [0,1]$ for $i = 1$ to n.

To circumvent the constraints on the w_i we represent each of the OWA weights as follows:

$$w_i = \frac{e^{\lambda_i}}{\sum_{j=1}^{n} e^{\lambda_j}}.$$

Under the above transformation it becomes clear that for any value of the parameters λ_i the weights w_i will lie in the unit interval and will sum to 1. Therefore the constrained minimization problem is transformed to the following unconstrained nonlinear programming problem:

Minimize the instantaneous errors e_k: (see equation at the bottom of the page) with respect to the parameters λ_i.

Inspired by the great success of gradient descent techniques in the back propagation method used for learning in neural networks [11] we shall use it here. Using the gradient descent method we obtain the following rule for updating the parameters $\lambda_i, i = (1, n)$

$$\lambda_i(l+1) = \lambda_i(l) - \beta \left.\frac{\partial e_k}{\partial \lambda_i}\right|_{\lambda_i = \lambda_i(l)}$$

where β denotes the learning rate $(0 \le \beta \le 1)$ and $\lambda_i(l)$ indicates the estimate of λ_i after the lth iteration.

For notational simplification we shall denote by $\hat{d}_k$ the estimate of the aggregated value d_k

$$\hat{d}_k = b_{k1}w_1 + b_{k2}w_2 + \cdots + b_{kn}w_n.$$

Then for the partial derivative $\partial e_k / \partial \lambda_i$ we get [8], [9]

$$\frac{\partial e_k}{\partial \lambda_i} = w_i(b_{ki} - \hat{d}_k)(\hat{d}_k - d_k), \qquad i = [1, n].$$

Finally, we derive the rule for updating the parameters λ_i as follows:

$$\lambda_i(l+1) = \lambda_i(l) - \beta w_i (b_{ki} - \hat{d}_k)(\hat{d}_k - d_k)$$

where the w_i are calculated at each iteration step from the currently estimated values of the λ_i

$$w_i = \frac{e^{\lambda_i(l)}}{\sum_{j=1}^{n} e^{\lambda_j(l)}}, \qquad i = [1, n]$$

and $\hat{d}_k$ is the current estimate of the aggregated values d_k.

The following summarizes the algorithm used at each iteration.

1) We have a current estimate of the $\lambda_i, \lambda_i(l)$ and a new observation consisting of the **ordered** arguments $b_{k1}, b_{k2}, \cdots, b_{kn}$ and an aggregated value d_k.
2) We use the $\lambda_i(l)$ to provide a current estimate of the weights
$$w_i(l) = \frac{e^{\lambda_i(l)}}{\sum_{j=1}^{n} e^{\lambda_j(l)}}.$$
3) We use the estimated weights along with the ordered arguments to get a calculated aggregated value
$$\hat{d}_k = b_{k1}w_1(l) + b_{k2}w_2(l) + \cdots + b_{kn}w_n(l).$$
4) We now update our estimates of the λ_i
$$\lambda_i(l+1) = \lambda_i(l) - \beta w_i(l)(b_{ki} - \hat{d}_k)(\hat{d}_k - d_k).$$

The following example illustrates the algorithm for learning OWA weights w_i.

Example 1: We assume the following collection of samples of data. Each sample consists of three argument values and a relevant aggregated value:

Sample	Argument Values			Aggregated Value
1	0.5	0.1	0.2	0.2
2	0.2	0.3	0.1	0.1
3	1.0	0.1	0.6	0.2
4	0.2	0.4	0.4	0.3
5	0.6	0.5	0.2	0.4

The learning algorithm was applied on the reordered argument tuples. The learning algorithm was started with initial values $\lambda_i(0) = 0, i = [1, 3]$, with initial values of the OWA weights $w_i = 0.33$. A learning rate of $\beta = 0.35$ was used. The estimated values of the λ_i after 350 iterations were

$$\lambda_1 = -2.16; \quad \lambda_2 = -1.61; \quad \lambda_3 = -0.46.$$

These values of the λ_i induce the following OWA weights:

$$w_1 = 0.121; \quad w_2 = 0.211; \quad w_3 = 0.668.$$

$$e_k = \tfrac{1}{2}\left(b_{k1}\frac{e^{\lambda_1}}{\sum_{j=1}^{n} e^{\lambda_j}} + b_{k2}\frac{e^{\lambda_2}}{\sum_{j=1}^{n} e^{\lambda_j}} + \cdots + b_{kn}\frac{e^{\lambda_n}}{\sum_{j=1}^{n} e^{\lambda_j}} - d_k\right)^2$$

Estimated aggregated values $\hat{d}_k$ at the end of the learning process were

$$\hat{d}_1 = 0.170; \quad \hat{d}_2 = 0.145; \quad \hat{d}_3 = 0.315;$$
$$\hat{d}_4 = 0.266; \quad \hat{d}_5 = 0.312.$$

IV. Induced OWA

As previously noted, the OWA aggregation is $F_w(a_1, \cdots, a_n) = W^T B$. Central to the implementation of the OWA aggregation is the step of ordering the arguments. This step produces the ordered argument vector B. The ordering used is based upon the value of the arguments, b_j is the value that is the jth largest of the arguments. Inspired by the work in [3], it appears that we can consider a more general policy toward the ordering of the arguments, the formulation of the ordered argument vector B.

We shall again assume that we have n values that we want to aggregate using the OWA weighting vector W, we denote these as $a_1, \cdots, a_n$, these are the arguments of the aggregation. In this more general framework, we shall assume each of the a_i values is a component of a more complex object which we shall for our immediate purpose represent as a two-tuple $\langle u_i, a_i \rangle$ and denote as an ***OWA pair.*** In this more general approach to OWA aggregation, we shall order the arguments, form the vector B, based upon the u_i values. In particular, our procedure for calculating the OWA aggregation of these OWA pairs

$$F_w(\langle u_1, a_1 \rangle, \cdots, \langle u_n, a_n \rangle) = W^T B_u$$

is as follows. We form the ordered argument vector B_u so that b_j is the a value of the OWA pair having the jth largest u value. In discussing these OWA pairs, $\langle u_i, a_i \rangle$, because of its role we shall refer to the u_i as the order inducing variable and a_i as the argument variable.

The following simple example illustrates the approach.

Example: Assume we have four OWA pairs $\langle u_i, a_i \rangle$ given

$$\langle 3, 0 \rangle, \langle 7, 0.2 \rangle, \langle 2, 0.9 \rangle, \langle 6, 1 \rangle$$

that we want to aggregate using the weighting vector

$$W = \begin{bmatrix} 0.4 \\ 0.3 \\ 0.2 \\ 0.1 \end{bmatrix}.$$

The first step is to order the OWA pairs, the $\langle u_i, a_i \rangle$, based upon the ordering inducing variable u_i. Performing this step gives us

Ordered OWA Pairs
$\langle 7, 0.2 \rangle$
$\langle 6, 1 \rangle$
$\langle 3, 0 \rangle$
$\langle 2, 0.9 \rangle$

From this order we obtain the components of the vector B by taking the ordered list of the a_i values thus

$$B = \begin{bmatrix} 0.2 \\ 1 \\ 0 \\ 0.9 \end{bmatrix}$$

that is

$$b_1 = 0.2, \quad b_2 = 1, \quad b_3 = 0 \quad \text{and} \quad b_4 = 0.9$$

Using this ordering, we get

$$F_w(\langle u_i, a_i \rangle) = W^T B = \sum_{j=1}^{4} w_j b_j$$
$$F_w(\langle u_i, a_i \rangle) = (0.4)(0.2) + (0.3)(1) + (0.2)(0) + (0.1)(0.9) = 0.47.$$

Let us look at the properties associated with these induced OWA (IOWA) operators, $F_w(\langle u_i, a_i \rangle)$. First it should be clear that these operators are communicative, the initial indexing of the pairs is unimportant, each of the objects involved in the aggregation is treated in the same way. We now show that the bounding property is exhibited by these IOWA operators. Assume that the ordered argument vector B induced by the order inducing variable is such that $\max_i[a_i] = b_q$. For any weighting vector W

$$F_w(\langle u_1, a_1 \rangle) = \sum_{j=1}^{n} w_j b_j \le \sum_{j=1}^{n} w_j b_q \le b_q \sum_{j=1}^{n} w_j \le b_q.$$

Now assume that $\min[a_j] = b_p$. For any weight vector W

$$F_w(\langle u_i, a_i \rangle) = \sum_{j=1}^{n} w_j b_j \ge b_p \sum_{j=1}^{n} w_j \le b_p.$$

Thus we see for any order inducing variable and any weighting vector

$$\min_i [a_j] \le F_w(\langle u_i, a_i \rangle) \le \max_j [a_j].$$

Directly following from this is the idempotency of these operators, if $a_i = a$ for all a_i then $b_i = a$ for all i and hence $\Sigma_{j=1}^{n} w_j bj = a$.

We now must show monotonicity. Assume that $\langle u_i, a_i \rangle, i = 1$ to n and $\langle u_i, i \rangle, i = 1$ to n are such that $a_i \ge \hat{a}_i$ for all i. Let B and $\hat{B}$ be the respective ordered argument vectors. Since the u_i are the same for both collections, then these two vectors are such that if $b_j = a_i$, then $\hat{b}_j = \hat{a}_i$. Since $a_i \ge \hat{a}_i$ for all i, then it can easily be seen that $W^T B \ge W^T \hat{B}$ and hence that

$$F_w(\langle u_i, a_i \rangle) \ge F_w(\langle u_i, \hat{a}_i \rangle)$$

if $a_i \ge \hat{a}_i$ for all i.

Thus we see that the aggregation of these OWA pairs are monotone. Based on the above we can see that these induced OWA operators can be viewed as mean aggregation operators with respect to the argument variable.

Implicit in the proof of monotonicity is the assumption that the order inducing variables are unchanged if this is not the case then monotonicity does not necessarily hold.

There are a number of ways in which the aggregation of OWA pairs is different from the aggregation of OWA singletons. Let $W^{[p]}$ be a weighting vector such that $w_p = 1$ and $w_j = 0$ for $j \neq p$, this vector has one in the pth position and zero elsewhere. What is clear is in the ordinary case, our aggregation is $\max_i[a_i]$ if $p = 1$ and is equal to the $\min_i[a_i]$ if $p = n$. In the case of the induced OWA operators, the max of the arguments is attained if p is the ordered position of the largest argument and the min is attained if p is the ordered position of the smallest of the arguments.

There exists an important distinction between the aggregation of OWA pairs and the ordinary OWA aggregation. This distinction arises when there is a tie in the ordering operation. Consider aggregation of the objects $\langle 5,1\rangle, \langle 3,0.5\rangle, \langle 8,0.6\rangle, \langle 5,0.4\rangle$ under the weighting vector

$$W = \begin{bmatrix} 0.4 \\ 0.3 \\ 0.2 \\ 0.1 \end{bmatrix}.$$

Performing the ordering of the objects we get

$$\underline{\textbf{Ordered OWA Pairs}}$$
$$\Rightarrow \quad \begin{matrix} \langle 8,0.6\rangle \\ \langle 5,1\rangle, \quad \langle 5,0.4\rangle \\ \langle 3,0.5\rangle \end{matrix}$$

We see that in this case, there is a tie between $\langle 5,1\rangle$ and $\langle 5,0.4\rangle$ with respect to order inducing variable. It can be easily shown that if we brake this tie by selecting $\langle 5,1\rangle$ ahead of $\langle 5,0.4\rangle$ giving us the ordered argument vector

$$B1 = \begin{bmatrix} 0.6 \\ 1 \\ 0.4 \\ 0.5 \end{bmatrix}$$

we would get a different aggregated value then by selecting $\langle 5,0.4\rangle$ ahead of $\langle 5,1\rangle$, which would us the ordered argument vector

$$B2 = \begin{bmatrix} 0.6 \\ 0.4 \\ 1 \\ 0.5 \end{bmatrix}.$$

That is

$$(0.4)(0.6) + (0.3)(1) + (0.2)(0.4) + (0.1)(0.5)$$
$$\neq (0.4)(0.6) + (0.3)(0.4) + (0.2)(1) + (0.1)(0.5).$$

The policy we shall follow in the case of ties in the order inducing process is to replace the arguments of the tied OWA pairs by the average of the arguments of the tied pairs in forming the B vector. Thus in the preceding illustration, when forming the B matrix we replace the argument component of each of $\langle 5,1\rangle$ and $\langle 5,0.4\rangle$ by their average 0.7, $(1+0.4)/2$. This substitution gives us an ordered argument vector

$$B = \begin{bmatrix} 0.6 \\ 0.7 \\ 0.7 \\ 0.5 \end{bmatrix}.$$

Following this process can be essentially shown to be equivalent to calculating the aggregated value as $W^T B = \frac{1}{2}[W^T B1 + W^T B2]$. We note if q items are tied, we replace these by q replica's of their average.

It should be clear that in the usual OWA aggregation ties don't present a problem, the reason for this is that the ordering variable is the same as the argument variable and however we place the tied objects leads to the same result.

V. Representations Using OWA Pairs

By appropriate selection of the order inducing variable many different types of argument aggregation can be modeled within this framework of induced OWA operators. In this section we shall look at some of these possibilities.

As a first example we show that if $u_i = a_i$, we obtain the usual OWA operator where the ordering in the B vector is induced by the value of the arguments. Consider $\langle u_i, a_i\rangle$ where $u_i = a_i$. Under our approach b_j is the a_i value corresponding to the jth largest u_i, however since $u_i = a_i$, then b_j is the a_i value of object having the jth largest a_i. Thus we see the regular OWA operator is a special case of the OWA pair aggregation when $u_i = a_i$.

We now show that the classic weighted average can be modeled in this framework of OWA pairs. We recall that in the classic weighted average aggregation, we have a collection of values $a_1, \cdots, a_n$ and an associated collection of weights $v_1, \cdots, v_n$ which lie in the unit interval and sum to one. In the case of the weighted averaging operator our aggregated value is calculated as

$$\text{Weighted Average} = \sum_{j=1}^{n} v_j a_j.$$

What is important to note here is that no ordering has taken place, a weight is directly associated with an argument. In particular the weight with index j, v_j, is directly associated with the argument a_j having index j. If V is the vector whose components are the v_j and A is the vector whose components are the a_j we can express this as $V^T A$. In order to model this weighted average aggregation within our IOWA framework using OWA pairs we proceed as follows. We form a weighting vector W such that $w_i = v_i$, that is $W = V$. We next form our OWA pairs as $\langle -i, a_i\rangle$, that is the argument value is a_i and the order inducing value is the negation of its index. Using these OWA pairs and the negation of the index i as the order inducing variable we get

$$\textbf{Ordered OWA Pairs}$$
$$\begin{matrix} \langle -1, a_1\rangle \\ \langle -2, a_2\rangle \\ \langle -3, a_3\rangle \\ \vdots \\ \langle -n, a_n\rangle \end{matrix}$$

From this we get as our ordered argument vector B where $b_i = a_i$, in particular we note that $B = A$. Since $F_w(\langle -i, a_i\rangle) = W^T B$ and we have $W = V$ and $B = A$ we get that

$$F_w(\langle -i, a_i\rangle) = V^T A = \sum_{j=1}^{n} v_j a_j.$$

Thus the weighted average is a special case of this more general technique of induced OWA operators.

It should be emphatically pointed out that in aggregating these OWA pairs, $\langle u_i, a_i\rangle$, we are only using the first element, the u_i values, to provide an ordering, the second components, the a_i, are the ones actually being aggregated. An important implication of this is that the values used for the u_i can be drawn from a space that is only required to have a linear ordering. This rather lax requirement on the u_i values allows us to use many different kinds of attributes for the order inducing variables. In particular, this characteristic allows us carry out aggregations that mix numbers with words in the aggregation, this can be seen as a step toward implementing Zadeh's paradigm of computing with words [4], [12]. The following example will provide a simple illustration of this feature of the IOWA operators.

Example: Consider a collection of OWA pairs $\langle u_i, a_i\rangle$ where the u_i are numbers and the u_i are values drawn from the space **£**. Let **£** be the space consisting of

$$£ = \{\text{very small } (VS), \text{small } (S), \text{medium } (M), \text{big } (B), \text{very big } (VB)\}$$

on which the following ordering exists on the objects

$$VB > B > M > S > VS.$$

Let $\langle S, 9\rangle, \langle VB, 3\rangle$, and $\langle M, 10\rangle$ be a collection of three OWA pairs we desire to aggregate using the weighting W where $w_1 = 0.3, w_2 = 0.5$, and $w_3 = 0.2$. Performing the ordering the OWA pairs with respect to the first component, we get

Ordered OWA Pairs

$$\langle VB, 3\rangle$$
$$\langle M, 10\rangle$$
$$\langle S, 9\rangle.$$

This ordering induces the ordered argument vector B,

$$B = \begin{bmatrix} 3 \\ 10 \\ 9 \end{bmatrix}$$

and from this we get an aggregated value

$$W^T B = (0.3)(3) + (0.5)(10) + (0.2)(9) = 7.7.$$

In some situations, when we use words for the u_i, we can use the implicit lexicographic ordering associated with words, this is the ordering that is used in listing words in dictionaries.

Example: Consider the following collection of OWA pairs $\langle \text{Johnson}, 160\rangle, \langle \text{Brown}, 70\rangle, \langle \text{Smith}, 20\rangle, \langle \text{Anderson}, 100\rangle$. Using a lexicographic ordering of this OWA pairs by the first argument we get

Ordered OWA Pairs

$$\langle \text{Anderson}, 100\rangle$$
$$\langle \text{Brown}, 70\rangle$$
$$\langle \text{Johnson}, 160\rangle$$
$$\langle \text{Smith}, 20\rangle.$$

This ordering induces the ordered argument vector

$$B = \begin{bmatrix} 100 \\ 70 \\ 260 \\ 20 \end{bmatrix}.$$

If the OWA weighting vector for this aggregation is

$$W = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}$$

we get as our aggregated value

$$W^T B = (0.1)(100) + (0.2)(70) + (0.3)(160) + (0.4)(20) = 80.$$

More generally, we see that if **£** is any set of objects such that there exists a linear ordering on **£**, for any distinct $y, x \in$ **£** then either $x > y$ or $x > y$ but not both, then we can draw our u values from **£**. We note that numeric values of course always satisfy this requirement.

One interesting class of models are those based on the nearest neighbor principle [13]. As we shall see our induced OWA operator allows for a generalization of these models. Assume we have a collection of n objects called patterns. Each pattern consists of two components, a locater or descriptor and prescribed value. For example, our objects may be people where the descriptor may some educational and physical characteristics of a person and the prescribed value is their salary. A problem of considerable interest is one in which we have some unclassified object, one for which we have descriptor information but no prescribed value and we desire to use the information contained in our patterns to provide information about the prescribed value of the unclassified object. One method often used is the nearest neighbor rule. Using this rule, we find the pattern closest to the unclassified object and then assign the prescribed value of this closest pattern to the unclassified object. We can formulate this approach within the framework of the IOWA operator. Each pattern of the n patterns, Pat(i), can be expressed as an pair, Pat$(i) = (L_i, V_i)$, where L_i is the location of this pattern and V_i is the prescribed value. We now have an unclassified object $\boldsymbol{x}$ whose location is $L_{\boldsymbol{x}}$ but whose prescribed value is unknown. For each of the given patterns, we calculate $u_i = -\text{Distance}[L_x, L_i]$, the negation of the distance between the pattern and the object $\boldsymbol{x}$. Using this, we form a collection of n OWA pairs, $\langle u_i, V_i\rangle$, one for each pattern. If we apply the OWA operator on these n OWA pairs, where our weighing vector W is such that $w_1 = 1$ and $w_j = 0$ for $j \neq 1$, and using

u_i as the order inducing variable and V_i as the argument we get the nearest neighbor rule. By selecting a W different from the one in which $w_1 = 1$ we can provide for a generalization of this nearest neighbor rule. For example, if $w_1 = w_2 = \frac{1}{2}$ and all other $w_j = 0$, then we get $V_{\boldsymbol{x}}$ as the average of the two closest neighbors. Other selections of W lead to different generalizations of the nearest neighbor principle. In using this to extend the simple nearest neighbor principle we impose the requirement on the weighting vector W that $w_i \geq w_j$ if $i < j$, this insures us that the closer neighbors have more influence. This approach using the IOWA operator becomes particularly useful when the distance between a pattern and the object to be classified can only be expressed on an ordered scale, very close, close, etc., for here we can take advantage of the IOWA operators ability to function in this ordinal environment.

Time indexed variables can also be represented in this framework of OWA pairs. Let $X(t)$ be a time indexed variable. We can express this as the OWA ordered pair $\langle u_i, a_i \rangle$ where $u_i = t$ and $a_i = X(t)$. Then we are ordering them with respect to line, the latest time is the first. If we use as our weighting vector

$$\begin{aligned} w_n &= (1-\alpha)^{n-1} \\ w_j &= \alpha(1-\alpha)^{j-1} \qquad j = 1, \cdots, n-1 \end{aligned}$$

we obtain the classic exponential smoothing [14]. If we use

$$\begin{aligned} W_j &= \frac{1}{m} \quad \text{for } j = 1 \text{ to } m \\ W_j &= 0 \quad \text{for all other } j \end{aligned}$$

we are taking the moving average of the last m readings.

VI. Modeling Using Induced OWA Operators

One goal of model construction is to be able to estimate the value of some variable, attribute of some object, based upon the known values of some other related variables which we shall call support variables. The construction of a model requires some knowledge about the domain which we are trying to model. Knowledge about an domain can come in many different forms. One kind of knowledge is raw observational data, other kinds of knowledge include experience or "expertise."

The procedure of constructing a model is usually divided into two phases, structure identification and parameter estimation. The structure identification phase, usually the more difficult, involves the selection of the support variables to included in the model and the formulation of the relationship between these variables. This formulation typically includes the introduction of some parameters. Thus the process of structure identification is essentially one of selecting a parameterized class of models and the determination of the support variables to be included. Mitchell [15] refers to this step of selecting a class of models as the introduction of an inductive bias. As he correctly indicates, if there is no assumption of a model, inductive bias, then there is no bases for evaluating the variable for unseen situations. The process of structural identification is usually based upon the use of "expertise" type knowledge rather than raw observation data and usually requires the intervention of some expert, typically human. However, we should note that clustering techniques which are based upon raw observational data, are being used to provide for automation of the structure identification step [16], [17]. We also note that data mining [18] techniques, again based upon observational data, are being used to help automate the identification step. Neural network methods also have been used to address the problem of structure identification. Here we can see that these techniques as a general movement in the direction of automating this process.

Once having dealt with the issue of structure identification, we begin the process of parameter estimation. This process, usually much easier to automate than structure identification, generally involves the use of raw observational data. It typically involves a process in which we estimate (learn) the parameters of the model by satisfying an optimization process in which our goal is to minimize the difference between what the model predicts and what we have observed. The error between the final model and the observations constitutes a measure of the performance of the model. Overall procedure may of course involve us returning to the structure identification step if we are not satisfied with the final model result. Implicit in this approach is what Mitchell [15] calls the inductive learning hypothesis, which essentially is a hypotheses saying "*any model found to provide a good approximation to the observations for a large training set will also provide a good approximation over other unobserved examples*." This, of course, may not hold.

The process of model building can most assuredly benefit from the availability of many different types of structures which can be used to help in the structure identification phase. Particularly useful are structures that can be used to represent expertise, this type of knowledge is often expressed in linguistic terms. One example of this is the fuzzy systems modeling technology [17], which has been very useful for modeling controllers.

The IOWA operators provide a class of relationships that can be used to help in process of model building. In these IOWA models, we have two types of support variables values, the u_j and the a_j. As we have seen, the a_j are those support variables that are actually combined to form the estimate of the variable of interest. The u_j are not actually aggregated but they play a role in assigning the weights to the a_j. Perhaps, a suitable analog to the role of the u_j can be seen in the antecedent condition of rules in rule based models [17]. In rule based system, the antecedent helps determine which rules are fired and then the consequents of these fired rules are aggregated, thus the a_j is analogous to the consequence.

In using the IOWA operator to model a variable the parameters associated with this model are the weights in the weighting vector. Thus the question of parameter estimation here becomes one of obtaining the components of the weighting vector. In preceding discussion on induced OWA operators, we implicitly assumed that the weighting vector W was available. It is natural to have to learn the weighting vector from examples. In the following, we shall see that the learning algorithm previously introduced for the ordinary OWA operator can be directly used to learn the components of the weighting vector when the objects being aggregated are OWA pairs.

Assume that we have a collection of m samples, each of these samples is comprised of n OWA pairs $O_{ki} = \langle u_{ki}, a_{ki}\rangle$, where k is the sample index and i is the pair index, thus O_{ki} is the ith pair in the kth sample. In addition, associated with each sample is a value d_k called the aggregated value of that sample. Our goal here is to find a weighting vector W such that the OWA aggregation of arguments for each sample, the a_{ki}, ordered by the u_{ki}, as best as possible matches the associated d_k. In particular, if B_k is the induced ordered vector for sample k, that is b_{kj} is the value of the pair having the jth largest u_{ki} in sample k, then we want to find W such that we minimize

$$\sum_{k=1}^{m} (W^T B_k - d_k)^2.$$

It should be clear that this is the same problem as we had before and therefore we can use the same procedure for learning the W. Thus our goal is to learn the w_i ($i = 1$ to n) which we express as $w_i = (e^{\lambda_i}/\Sigma_{j=1}^n e^{\lambda_j})$. The following is the algorithm.

1) We have a current estimate $\lambda_i(l)$ of λ_i and an observation consisting of the OWA pairs $\langle u_{k1}, a_{k1}\rangle, \langle u_{k2}, a_{k2}\rangle, \cdots, \langle u_{kn}, a_{kn}\rangle$ and the value d_k. We order the objects with respect to the u_{ki} to give us B_k where b_{kj} is the a_{ki} corresponding to the jth largest U_{ki}.
2) We used $\lambda_i(l)$ to obtain a current estimate of the weighting vector $W(l)$ where
$$w_i(l) = \frac{e^{\lambda_i(l)}}{\sum_{j=1}^{n} e^{\lambda_j(l)}}.$$
3) We calculate $\hat{d}_k = W(l)^T B_k$.
4) We update our estimate of λ_i using the following:
$$\lambda_i(l+1) = \lambda_i(l) - \beta w_i(l)(b_{ki} - \hat{d}_k)(\hat{d}_k - d_k).$$

The important thing to note is that we have a technique available to use for learning the OWA weighting vector in situation in which we are aggregating OWA pairs, thus we can learn in IOWA models.

VII. Best Yesterday Model

In the following, we give an example of the use of the induced OWA operator in model building. We shall see that the imperative used here, which is a kind "best yesterday" approach, is very suitable to the use of induced OWA operators.

Our problem is to estimate tomorrows opening price of the stock of the XYZ Company. In order to obtain this opening price, we use four experts. Each of these experts provides a prediction of the opening price on the night before. We are now faced with the problem of ***fusing*** these individual expert's predictions to obtain our desired estimate of the opening price of the XYZ stock. Consider Table I.

One approach to obtaining a fused prediction from those of experts A, B, C, and D is to use a linear regression model. In this case, we have that P_k, predicted opening price on day k, is expressed as

$$P_k = W_A X_A(k) + W_B X_B(k) + W_C X_C(k) + W_D X_D(k).$$

In this model, $X_A(k)$ is the estimate provided by expert A for day k and W_A is the weight assigned to expert A. Using this model and the usual least square estimation procedure, we get

$$W_A = 0.84, \quad W_B = 0.16, \quad W_C = 0.55, \quad W_D = -0.42.$$

With this model, given a prediction by the four experts, we calculate our predicted value using the formula. Using these weights, we get as our forecasts for the opening price on day k

$$P_1 = 107.25, \quad P_2 = 122.45, \quad P_3 = 88.37,$$
$$P_4 = 116.18, \quad P_5 = 94.75.$$

The rms error of this approximation to the opening price is 5.11.

We now consider another structure to model the opening price of the stock. This model which makes use of the induced OWA aggregation method implements a kind of best yesterday imperative. In this model, our predicted opening price $\hat{P}(k)$ for day k is obtained as

$$\hat{P}(k) = w_1 y_1(k) + w_2 y_2(k) + w_3 y_3(k) + w_4 y_4(k).$$

In this model $y_j(k)$ is the day k prediction of the expert who had the jth best prediction on day $k-1$, yesterday, and w_j is the weight we give to the expert who had the jth best prediction yesterday. In this model, as opposed to the usual linear regression model, the weights are not associated with a particular expert, but are associated with how well they did yesterday. Let us more formally represent this model. We shall let $\hat{P}(k)$ be what we are interested in predicting, the opening price of the stock for day k. We shall let $X_A(k), X_B(k), X_C(k)$, and $X_D(k)$ be the predictions of the respective experts. We shall let $TP(k-1)$ be the actual opening price on day $k-1$, which is already known at the point where we are making our prediction for day k. In addition we let $X_A(k-1), X_B(k-1), X_C(k-1)$, and $X_D(k-1)$ be the predictions of the various experts for day $k-1$. Using this information we can construct an induced OWA model to be used to aggregate the individual experts predictions. For day k we associate with each expert an OWA pair $\langle u_A(k), a_A(k)\rangle, \langle u_B(k), a_B(k)\rangle, \langle u_C(k), a_C(k)\rangle, \langle u_D(k), a_D(k)\rangle$ where

$$\begin{aligned}
u_A(k) &= -|TP(k-1) - X_A(k-1)| \\
a_A(k) &= X_A(k) \\
u_B(k) &= -|TP(k-1) - X_B(k-1) \\
a_B(k) &= X_B(k) \\
u_C(k) &= -|TP(k-1) - X_C(k-1)| \\
a_C(k) &= X_C(k) \\
u_D(k) &= -|TP(k-1) - X_D(k-1)| \\
a_D(k) &= X_D(k).
\end{aligned}$$

TABLE I
EXPERT DATA

	Expert Predictions				Actual
Day #	A	B	C	D	Opening Price
1	101	99	82	116	100
2	97	76	90	121	92
3	96	75	91	107	100
4	104	95	90	118	99
5	105	89	91	112	105

That is the u value for each expert is the negation of the absolute difference between the experts prediction for yesterday and the actual open price yesterday. We note because of the use of negation, the larger the u value the better the experts performance yesterday. The a value is of course the experts prediction for today. Associated with this model we have an OWA weighting vector

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}.$$

Using the OWA pairs described above we calculate the prediction for day k as

$$\hat{P}(k) = F_w(\langle u_A(k), a_A(k)\rangle, \langle u_B(k), a_B(k)\rangle \\ \langle u_C(k), a_C(k)\rangle, \langle u_D(k), a_D(k)\rangle)$$

which is the OWA aggregation of the OWA pairs. Using this structure we get as our predicted value as desired

$$\hat{P}(k) = w_1 y_1(k) + w_2 y_2(k) + w_3 y_3(k) + w_4 y_4(k)$$

where $y_j(k)$ is today's prediction of yesterday jth best expert. We note a fundamental distinction between this and the linear regression model is that the weights, the w_i are not directly linked to an expert but linked to position of performance yesterday. If, for example, W is such that $w_1 = 1$ and $w_j = 0$ for $j \neq 1$ then our fused prediction for tomorrow is that of the expert who performed best yesterday. If W is such that $w_1 = w_2 = w_3 = w_4 = 0.25$, we take the simple average of all the experts.

In the framework of model construction by deciding to use the IOWA model with the selected form for the OWA pairs we have essentially implemented the structure identification phase. In this model the parameter we must now estimate is the weighting vector W. To accomplish this we can use the algorithm we described earlier for learning the weights in the face of OWA pairs.

Using the same data we used previously (see Table I) we obtain the ranking of the experts for each day from the past day and applying the learning model which we previously discussed, we obtain the OWA weights

$$w_1 = 0.20, \quad w_2 = 0.12, \quad w_3 = 0.08, \quad w_4 = 0.6.$$

Using these weights in our model we obtain as our estimates of the opening prices

$$\hat{P}(1) = 100.07, \quad \hat{P}(2) = 92.14, \quad \hat{P}(3) = 100, \\ \hat{P}(4) = 99 \quad \text{and} \quad \hat{P}(5) = 105.$$

The rms error of this model is 0.1936, thus we see in this case, the IOWA model is better than the regression model.

While we have used the experts performance yesterday as the order inducing variable, it is possible to use other forms for order inducing variable. For example, we could use

$$u_i(k) = -\tfrac{1}{2}\,(|TP(k-1) - X_i(k-1)| \\ + |TP(k-2) - X_i(k-2)|).$$

In this case, the order inducing value for expert i is based upon the experts performance for the last two days. More generally, we can use

$$u_i(k) = -\frac{1}{m}\left|\sum_{j=1}^{m} |TP(k-j) - X_i(k-j)|\right..$$

Here we are taking the average of experts performance for the last m days. Actually, we can suggest an even more general function to estimate $u_i(k)$, the order inducing variable. Let

$$\Delta_i(k,j) = |TP(k-j) - X_i(x-j)|$$

the absolute error of expert i, j days ago. Using this, one possibility for $u_i(k)$ is

$$u_i(k) = -\sum_{j=1}^{m} q_j \Delta_i(k,j)$$

where the q_j are a set of normalized positive weights.

We can also express $u_i(k)$ using an OWA operator, let G_W be an OWA operator of dimension m with weighting vector W then

$$u_i(k) = -G_w(\Delta_i(k,1), \Delta_i(k,2), \cdots, \Delta_i(k,m)).$$

VIII. CONCLUSION

In this work we introduced a new extension of the OWA operator called the Induced Ordered Weighted Averaging (IOWA) Operator. These aggregation operators take as their input OWA pairs in which one component induces a ordering over the second components, the argument value, which are then aggregated. A number of different aggregation situations have been shown to be representable in this framework. In many areas such as decision making [19], sensor fusion [20], distributed detection theory [21], and social welfare modeling [22] there is a need to fuse information, it appears that these IOWA operators may be able to help in addressing this problem.

References

[1] R. R. Yager, "On ordered weighted averaging aggregation operators in multi-criteria decision making," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 183–190, 1988.

[2] R. R. Yager and J. Kacprzyk, *The Ordered Weighted Averaging Operators: Theory and Applications.* Norwell, MA: Kluwer, 1997.

[3] H. B. Mitchell and D. D. Estrakh, "A modified OWA operator and its use in lossless DPCM image compression," *Int. J. Uncertainty, Fuzziness, Knowledge Based Syst.*, vol. 5, pp. 429–436, 1997.

[4] L. A. Zadeh, "Fuzzy logic = computing with words," *IEEE Trans. Fuzzy Syst.*, vol. 4, pp. 103–111, 1996.

[5] R. R. Yager, "Families of OWA operators," *Fuzzy Sets Syst.*, vol. 59, pp. 125–148, 1993.

[6] ——, "On the inclusion of importances in OWA aggregations," in *The Ordered Weighted Averaging Operators: Theory and Applications*, R. R. Yager and J. Kacprzyk, Eds. Norwell, MA: Kluwer, 1997, pp. 41–59.

[7] M. O'Hagan, "Aggregating template or rule antecedents in real-time expert systems with fuzzy set logic," in *Proc. 22nd Annu. IEEE Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, 1988, pp. 681–689.

[8] D. P. Filev and R. R. Yager, "Learning OWA operator weights from data," in *Proc. 3rd IEEE Int. Conf. Fuzzy Systems*, Orlando, FL, 1994, pp. 468–473.

[9] ——, "On the issue of obtaining OWA operator weights," *Fuzzy Sets Syst.*, vol. 94, pp. 157–169, 1998.

[10] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA, 1974.

[11] ——, *The Roots of Backpropagation*. New York: Wiley, 1994.

[12] L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy Sets Syst.*, vol. 90, pp. 111–127, 1997.

[13] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[14] R. G. Brown, *Smoothing, Forecasting and Prediction of Discrete Time Series*. Englewood Cliffs, NJ: Prentice-Hall, 1963.

[15] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[16] R. R. Yager and D. P. Filev, "Generation of fuzzy rules by mountain clustering," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 209–219, 1994.

[17] ——, *Essentials of Fuzzy Modeling and Control*. New York: Wiley, 1994.

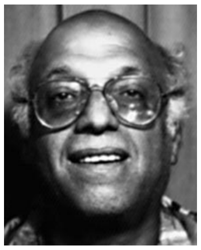
[18] G. Piatetsky-Shapiro and B. Frawley, *Knowledge Discovery in Databases*. Cambridge, MA: MIT Press, 1991.

[19] R. D. Luce and N. Raiffa, *Games and Decisions: Introduction and Critical Survey*. New York: Wiley, 1967.

[20] E. Waltz and J. Llinas, *Multisensor Data Fusion*. Norwood, MA: Artech House, 1990.

[21] A. Pete, K. R. Pattipati, and D. L. Kleinman, "Distributed detection in teams with partial information: A normative-descriptive model," *IEEE Trans. Syst., Man, Cybern,*, vol. 23, pp. 1626–1648, 1993.

[22] J. Kacprzyk, H. Nurmi, and M. Fedrizzi, *Consensus Under Fuzziness*. Boston, MA: Kluwer, 1997.



Ronald R. Yager (S'66–M'68–SM'93–F'97) received his undergraduate degree from the City College of New York and the Ph.D. degree from the Polytechnic University of New York, Brooklyn.

He is Director of the Machine Intelligence Institute and Professor of Information Systems, Iona College, New Rochelle, NY. He has served at the National Science Foundation as Program Director in the information sciences program. He was a NASA/Standard fellow. He was a Research Associate at the University of California, Berkeley. He was a Visiting Scholar at Hunter College, City University of New York. He has served as a Lecturer at the NATO Advanced Study Institutes. He is a Research Fellow of the Knowledge Engineering Institute, Guangzhou University, China. He is on the scientific committee of the Fuzzy Logic Systems Institute, Iizuka, Japan. He is Co-President of the International Conference on Information Processing and Management of Uncertainty, Paris. He is Editor-in-Chief of the *International Journal of Intelligent Systems*. He also serves on the editorial boards of a number of other journals, including *Neural Networks*, *Data Mining and Knowledge Discovery*, the *Journal of Approximate Reasoning*, *Fuzzy Sets and Systems* and the *International Journal of General Systems*. He has published more than 450 articles and 15 books. His current research interests include information fusion, intelligent agent technologies, multi-criteria decision making, information retrieval, multimedia information systems, knowledge discovery, uncertainty management, fuzzy set theory, fuzzy-neural modeling, and uncertainty in databases.

Dr. Yager is a fellow of the New York Academy of Sciences and the International Fuzzy Sets Association.

Dimitar P. Filev (M'95–SM'97) received the Ph.D. degree in electrical engineering from the Czech Technical University, Prague, Czech Republic, in 1979.

He is a Senior Technical Specialist with Ford Motor Company, Detroit, MI. Prior to joining Ford, he was an Associate Professor of information systems and Senior Research Associate at the Machine Intelligence Institute, Iona College, New Rochelle, NY, and Senior Research Associate at Central Lab for Bioinstrumentation and Automation, Bulgarian Academy of Science. He is conducting research in control theory and applications, modeling of complex systems, fuzzy modeling, and control. He has published over 130 articles in refereed journals and conference proceedings and co-authored three books.

Dr. Filev was a recipient of the 1995 Award for Excellence from MCB University Press and the 1996 Henry Ford Technology Award. He is a Associate Editor of the IEEE Transactions on Fuzzy Systems.