Mathware & Soft Computing 6 (1999) 249-265

On the Learning of Weights in Some Aggregation Operators: the Weighted Mean and OWA Operators

V. Torra
Institut d'Investigació en Intel·ligència Artificial - CSIC
Campus UAB s/n, 08193 Bellaterra (Catalunya, Spain)
vtorra@iiia.csic.es

Abstract

We study the determination of weights for two types of aggregation operators: the weighted mean and the OWA operator. We assume that there is at our disposal a set of examples for which the outcome of the aggregation operator is known. In the case of the OWA operator, we compare the results obtained by our method with another one in the literature. We show that the optimal weighting vector is reached with less cost.
Keywords: Learning weights, Modeling, Aggregation operators, Weighted Mean, OWA operators, WOWA operators

1 Introduction

Aggregation operators are used nowadays in several applications in Artificial Intelligence to combine information from different information sources (opinions of experts, data from sensors, results from computational systems). We find their implementation in several systems. For example: vision [López et al., 1994], robotics [López de Mántaras et al., 1997], knowledge based systems [Torra et al., 1995]. Some examples of well known aggregation methods are the weighted mean (characterized in [Aczél, 1984, Aczél et al., 1986]), the OWA operator (defined in [Yager, 1993, 1988] and characterized in [Fodor et al., 1995]) and the Choquet integral (see [Grabisch et al., 1995, Fodor et al., 1994] for a review of fuzzy integrals). See [Grabisch et al., 1995, Torra, 1998] for a review of some aggregation operators.

Although these operators have already been used in several applications, in order to apply them to new problems users have to fix the parameters that are associated with each of the methods. For example, to use a weighted mean or an OWA operator the user has to settle the weights of each information source; and to use a Choquet integral the definition of a fuzzy measure is required. The determination of these weights is usually done in an heuristic way (after trial and error) or asking an expert to supply them.

In this work we present a solution to this problem based on a machine learning approach and using optimization techniques. We assume that there is a set of examples at our disposal, and that we are interested in learning a model once the aggregation operator has already been decided. For us, an example consists on a set of values corresponding to the values supplied by information sources (the input values) and the resulting value after the aggregation (the output value). Learning the model consists on determining the parameters to be used in the aggregation function. For example, in the case of the weighted mean, finding the model consists on determining the weights of each information source.

In this work we consider the learning of weights for several aggregation operators. We consider the weighted mean and the OWA operator. We also give some comments on the process of learning the weights for the WOWA operator although no example is provided in this case.

The structure of this work is as follows. In Section 2 we review the aggregation operators that are used in the rest of the work. Section 3 discusses the formal aspects related to weight learning and Section 4 describes some examples. The article finishes in Section 5 with the conclusions.

2 Preliminaries

We define below the aggregation functions that we consider in this work: the weighted mean (studied and characterized in [Aczél, 1984, Aczél et al., 1986]), the OWA (Ordered Weighting Averaging) operator defined in [Yager, 1993, 1988]), and the WOWA (Weighted OWA) operator.

Definition 1. A vector $\mathbf{v}=[v_1\ v_2\ \ldots\ v_n]$ is a *weighting vector* of dimension n if and only if

$$v_i \in [0,1] \quad \Sigma_i v_i = 1$$

Definition 2. Let $\mathbf{p}$ be a weighting vector of dimension n, then a mapping WM: $\mathbb{R}^n \to \mathbb{R}$ is a *weighted mean* of dimension n if $WM_p\ (a_1,\ldots,a_n) = \Sigma_i\ p_i\ a_i$.

Definition 3 [Yager, 1988, 1993]. Let $\mathbf{w}$ be a weighting vector of dimension n, then a mapping OWA_w: $\mathbb{R}^n \to \mathbb{R}$ is an *Ordered Weighted Averaging (OWA) operator* of dimension n if

$$OWA_w(a_1,\ldots,a_n) = \Sigma_i w_i a_{\sigma(i)}$$

where $\{\sigma(1),\ldots,\sigma(n)\}$ is a permutation of $\{1,..,n\}$ such that $a_{\sigma(i-1)} \geq a_{\sigma(i)}$ for all i=2, ..., n. (i.e., $a_{\sigma(i)}$ is the i-th largest element in the collection $a_1,\ldots, a_n$).

As these definitions underline, both operators are linear combination of the values according to a set of weights. Their difference is the fact that in the case of the OWA operator an ordering of the values is performed prior to the combination. This difference forces a different meaning between the two functions. On one hand, the weighted mean allows the system to compute an aggregate value from the ones corresponding to several sources taking into account the reliability of each information source. This corresponds to measure the importance of information sources. Alternatively, the OWA operator permits of weighting the values in relation to their ordering. This corresponds to give importance to values instead of sources. See [Torra, 1997] for a detailed comparison of both aggregation functions.

In [Torra, 1997] it was defined the WOWA (Weighted OWA) operator. It is an aggregation function that allows to consider the two kind of weights introduced above (importance of sources and importance of values). This is, the weights corresponding to the weighted mean and the weights corresponding to the OWA operator.

Definition 4. Let **p** and **w** be two weighting vectors of dimension n, then a mapping WOWA:$\mathbb{R}^n \rightarrow \mathbb{R}$ is a *Weighted Ordered Weighted Averaging (WOWA) operator* of dimension n if

$$\mathrm{WOWA}_{p,w}(a_1, ..., a_n) = \Sigma_i \omega_i a_{\sigma(i)}$$

where $\{\sigma(1), ..., \sigma(n)\}$ is a permutation of $\{1,..,n\}$ such that $a_{\sigma(i-1)} \geq a_{\sigma(i)}$ for all i=2, ..., n. (i.e., $a_{\sigma(i)}$ is the i-th largest element in the collection a_1,..., a_n), and the weight ω_i is defined as

$$\omega_i = w^*(\Sigma_{j \leq i} p_{\sigma(j)}) - w^*(\Sigma_{j<i} p_{\sigma(j)})$$

with w^* a monotone increasing function that interpolates the points (i/n, $\Sigma_{j \leq i}$ w_j) together with the point (0,0). The function w^* is required to be a straight line when the points can be interpolated in this way. One interpolation method suitable for building the function w^* is described in [Torra, 1999]. From now on, ω represent the set of weights $\{\omega_i\}$, i.e., $\omega = [\omega_1, ..., \omega_n]$.

3 Learning weights from examples

In this section we study the process of learning the weights of the aggregation operators described in the previous section. We consider a set of examples, each consisting on a set of values to aggregate and the outcome of the aggregation. From now on, we assume that the number of examples at our disposal is M and that the dimension of the weighted mean is settled to N. This latter condition means that we consider examples where the number of information sources (the number of criteria or elements) to combine is N. According to this, and as for each example we assume that an approximate value of the outcome of the aggregation function is known, the structure of the examples is the one given in Table 1. In this table, a^i_j corresponds to the value supplied by the j-th source in the i-th example; and m^i corresponds to the outcome of the i-th example.

a^1_1	a^1_2	a^1_3	...	a^1_N	m^1
a^2_1	a^2_2	a^2_3	...	a^2_N	m^2
...					
a^M_1	a^M_2	a^M_3	...	a^M_N	m^M

Table 1. Data examples

3.1 Learning weights in the weighted mean

To find the adequate set of weights, we need to formalize this definition. To do so, we define the best model as the one that minimizes the accumulated distance for all examples, using as a distance the squared difference (i.e., distance(x,y)=(x-y)^2). Under this condition, we have that the best model is the one that minimizes:

$$D(\mathbf{p}) = \sum_{j=1}^{M} \left(\sum_{i=1}^{N} WM_p(a) - m^j \right)^2$$

This distance, according to the definition of the weighted mean is equivalent to:

$$D(\mathbf{p}) = \sum_{j=1}^{M} \left(\sum_{i=1}^{N} a_i^j p_i - m^j \right)^2 \quad (1)$$

Using the Euclidean norm $||\mathbf{x}|| = \sqrt{(\mathbf{x}'\mathbf{x})}$, we can express D(p) alternatively as D(p)= $||H\ p - d\ ||^2$. Here H stands for the set of measurements obtained from the information sources, this is H=$\{a_i^j\}$, and d stands for the "ideal" solutions that should give the system. This is, d=(m^1, ..., m^M). Note also that we use A' to denote the transpose of A. To obtain a solution we have to minimize this distance. This corresponds to the method of linear least squares [Stoer et al., 1980].

However, expressing the distance in this way and minimizing it is not enough to get a correct weighting vector. We have to minimize this expression, but in order that the solution p is a weighting vector, we need weights p_i to be positive and add to one. These two conditions can be included in the problem resulting a minimization problem with constraints:

$$\text{Min } D(\mathbf{p}) \quad (2)$$

such that

$$\sum_{i=1}^{N} p_i = 1 \quad (3)$$

$$p_i \geq 0 \text{ for all i} \quad (4)$$

Using the expression D(p) = $||H\ p - d\ ||^2$ = (H p - d)(H p - d) and the fact that this expression gets its minimum when we have the minimum of (1/2) p' H' H p - H' d p, we can reformulate this problem in the following way (where Q = H' H and c = - H' d).

$$\text{Min } (0.5)p'Qp + c'p \quad (2)$$

such that

$$\sum_{i=1}^{N} p_i = 1 \quad (3)$$

$$p_i \geq 0 \text{ for all } i \tag{4}$$

The problem formulated in this way is a typical optimization problem: a quadratic program with inequality constraints. There exists several algorithms to solve this problem (see, for example, [Luenberger, 1973, Gill et al., 1981]). In particular, it can also be solved using active set methods.

These methods exploit the fact that the computation of the solution of quadratic problems with linear equality constraints is simple. Active set methods are iterative ones in which at each step inequality constraints are partitioned into two groups: those that are to be treated as active (they are considered as equality constraints) and inactive (they are essentially ignored). Once the partition is known, the algorithm proceeds moving on the surface defined by the working set of constraints (the set of active constraints) to an improved point. In this movement some constraints are added to the working set and some other ones are removed. Then, the algorithm computes a new movement on the surface. This process is repeated until the minimum is reached.

In our case, inequality constraints are the ones that restrict the weights to be positive. According to this, when the restriction corresponding to the weight p_i is active the weight is forced to be zero. Instead, when a restriction is not active the value of the corresponding weight is not restricted. Thus, at a certain step, the working set is defined by the initial equality constraint (all weights add to one) and the active ones.

Assuming that at the k-th step, p_k is a non-optimal solution found in the previous step, the movement on the surface described above consists on modifying p_k so that better approximation (more minimal than the previous one) is obtained. This is, at time k a vector d_k is computed that corresponds to the step to perform from the last solution p_k. As, we do not want to violate the constraints in the working set when we compute $p_{k+1}= p_k + d_k$ we require that d_k does not modify the constraints and thus $\Sigma d_i = 0$. So, we require that all equality constraints in the working set are zero.

The vector d_k is obtained as the solution of the following optimization problems:

$$\begin{array}{rl} \text{minimize} & (1/2)d_k' Q d_k + g_k' d_k \\ \text{subject to} & a_i' d_k = 0 \text{ for all } i \in W_k. \end{array}$$

where g_k=c + Q p_k and a'$_i$ are the coefficients of equality constraints in the working set W_k.

This problem can be solved with the following system of linear equations [Luenberger, 1973]:

$$\begin{array}{rcl} Qd_k + A'\lambda & = & -g_k \\ Ad_k & = & 0 \end{array} \tag{5}$$

where λ are the Lagrange multipliers, and A is the matrix formed with all the coefficients of the active constraints.

To complete the method we need to determine the procedures for adding a constraint to the working set and to remove a constraint. Both aspects are given below in a version of the whole algorithm. The algorithm needs an initial weighting vector that satisfies all the constraints. This is easy to find as p=(1/n, .., 1/n), being n the number of weights, can be used.

```
procedure Learning_Weights_WM
                                        (p: weighting vector,
                                        Q: quadratic matrix,
                                        c: vector) returns weighting vector is
begin
    k = 0; $p_k$ = p;
    $W_k$ = the equality constraint corresponding to have a=(1, ..., 1);
    exit := false;
    while not exit do
      $g_k$ = c + Q $p_k$;
      compute $d_k$ and $\lambda$ (lagrange multipliers) as a solution of:
        minimize (1/2) $d'_k$ Q $d_k$ + $g'_k$ $d_k$
        subject to $a'_i$ $d_k$ = 0 for all i $\in W_k$.
      if $d_k \neq 0$ then
      $\alpha_k$ = min {1, min{ ($b_i$ - $a'_i$ $p_k$)/($a'_i$ $d_k$) // $a'_i$ $d_k > 0$}};
      $p_{k+1}$ = $p_k + \alpha_k d_k$;
      if $\alpha_k < 1$ then
      - -add restriction (the equality constraint) corresponding to $p_{\alpha k} = 0$
       $W_{k+1}$= $W_k$ + the restriction of the minimizing index of $\alpha_k$;
       check-lagrange := false;
      else check-lagrange := true;
      end if;
      else
      check-lagrange := true;
      end if;
      if check-lagrange then
      $\lambda_q$ = min {$\lambda_i$ // i $\in$ I $\cap$ $W_k$};
      if $\lambda_q$=0 then exit:=true;
      else
      - - drop restriction (equality constraint) corresponding to q-th weight
        $W_{k+1}$ = $W_k$ - the q-th weight restriction;
      end if;
      end if;
      k:=k+1;
    end while;
    return $p_k$;
end procedure;
```

3.2 Aspects related to the optimization problem

When applying optimization methods to learn weights from examples, we are assuming that the problem to minimize has either a single solution or all the solutions are equally adequate for the problem. Moreover, several optimization methods, as the ones based on active sets, assume that the matrix Q that define the problem is nonsingular (a square matrix whose columns are linearly independent). As both aspects are not always true in relation to the process of learning weights for an arithmetic mean we consider them in more detail:

1. All solutions (weighting vectors) are not equally adequate.

This is so because usually we have that information sources (experts or sensors) are redundant. In this case, several different solutions with the same D(p) can be obtained. In this case, it is better to distribute weights among all the sources than to have the weights accumulated in a single information source. If p_1=(0.5, 0.5) and p_2=(0,1) and D(p_1)=D(p_2) then p1 is a better solution because the final aggregated value would be less influenced by the error of a single source. In other words, it is always preferable that the weights are as much distributed (or dispersed) as possible.

2. Quadratic problems can have singular matrices.

In general, if we have redundant information sources, we have that the output of some of them can be deduced from the values given by some of the other sources. In this case, the matrix Q is singular. We will study this case with more detail below.

In relation to the singularity of matrix Q, we can state the following:

1. If the columns of H are linearly independent, then the matrix Q = H'H is nonsingular (see e.g., [Stoer et al., 1980]). In this case, the least squares problem has a single solution. When, we combine this minimization problem with the restrictions, there is still a single best solution. Linear systems used in active set methods to solve the minimization problem have single solutions and are solved using a non-singular matrix.

2. When the matrix H has columns that are linealy dependent, then Q is singular. However, some problems have single solutions.

We establish these results in the following propositions. We begin considering the case in which the columns in H are linearly independent.

Proposition 1 [Luenberger, 1973, p.424]. Let Q and A be n * n and m * n matrices, respectively. Suppose that A has rank m and that Q is positive definite on the subspace M = {x : A x = 0}. Then the matrix

$$\begin{pmatrix} Q & A' \\ A & 0 \end{pmatrix}$$

is nonsingular.

Proposition 2. Let H, the examples, be a matrix of dimension M * N (number of examples * number of columns). In this case, if the columns of H are linearly independent, there is a single weighting vector **p** that minimizes the distance D(**p**).

Proof. First of all, the matrix Q = H'H is nonsingular [Stoer et al., 1980], definite positive and of rank N. Second, for all working set W with M$\leq$N equations, its corresponding matrix A of dimension M*N is of rank M. Then, using what it is stated in proposition 1, there is a single solution for the system (5) at each step. As it is not possible to obtain for two working sets, two minimal solutions with the same D(p) (unless it is the same p) the solution is unique. Note that as the optimization problem is convex, two different minimal solutions would define a convex surface and thus some systems (5) would not have a single solution. □

Therefore, when there are no columns in H that are linear combinations, there is a single solution of the minimization problem. We consider now the case in which there are columns in H that are linearly dependent.

Proposition 3. If there is a column k in H that is a linear combination of the other ones in H, then the matrix Q = H'H has column k and row k (note that Q is simmetric) that are linear combinations of the other ones, also, the vectors c=-H'd and g_k=c+Q p_k have the k-th element that is a linear combination of the other ones.

In this case, the minimization problem subject to constraints can have several solutions with the same minimization value. This is the case when the column c_k is a linear combination of other columns in such a way that

$$c_k = \sum_{i \neq k} \alpha_i c_i$$

with $\sum_i \alpha_i = 1$ for $\alpha_i \geq 0$.

In this case, although there are several solutions, the following proposition holds.

Proposition 4. If there is a column c_k in H that is a linear combination of the other columns c_i in such a way that:

$$c_k = \sum_{i \neq k} \alpha_i c_i$$

with $\sum_i \alpha_i = 1$ for $\alpha_i \geq 0$, then if the vector p is a solution of the minimization of D(p) with p such that (3) and (4) hold and $p_k \neq 0$, then there is, at least, another vector p' such that D(p')=D(p) such that (3) and (4) also hold.

Proof. The weighting vector p' defined by $p'_i = p_i + \alpha_i p_k$ for all i$\neq$k, and p'_k=0 is such that D(p')=D(p). Note that, as $\sum \alpha_i = 1$, it holds $\sum_i p'_i = 1$ and $p_i \geq 0$. Therefore p' is a weighting vector. Using the results stated in the last proposition it is straightforward to prove D(p)=D(p'). □

This proposition implies that we can eliminate the k-th column in the matrix H, and solve the minimization problem without it. The minimal D(p) will be the same.

Proposition 5. If there is a column c_k in H that is a linear combination of the other columns c_i in such a way that:

$$c_k = \sum_{i \neq k} \alpha_i c_i$$

with $\sum_i \alpha_i = 1$ for $\alpha_i \geq 0$, then the column c_k can be eliminated from H and the minimal value D(p) does not change.

Proof. Let p be a solution using the initial H. Note that if $p_k \neq 0$ then, according to the previous proposition, there exists a different p' such that D(p')=D(p) with p_k=0. Therefore, the k-th column can be eliminated and the minimum value D(p) does not change. Alternatively, if p_k=0 then D(p) is also the same when the column c_k is not used. □

In a general case, however, it could be the case that no column c_k could be written as a linear combination of the other columns with $\sum \alpha_i \neq 1$. In this case, it is possible that the minimization problem has still a single solution. This is so because in (5) we need Q to be definite positive but only on the subspace M={x: Ax=0}. Let us consider the following example:

Example 1. Let us learn the weights from a single observation H=(1 3) and d=(1). In this case, the matrix Q and the vector c are defined as follows:

$$Q = \begin{pmatrix} 1 & 3 \\ 3 & 9 \end{pmatrix} \qquad c = \begin{pmatrix} -1 \\ -3 \end{pmatrix}$$

In this case, we have that c_1=c_2/3 and thus the initial conditions in the previous proposition do not apply. In this example there is a single vector p that minimizes D(p). The solution is p=(1 0).

In fact, this problem can be solved by active set methods because, although the matrix Q is singular, when the constraint corresponding to the weights is added, it is possible to find a solution. If we consider the system of equations in (5) for this problem, with p_k=(1/2, 1/2), d_k=(x_1, x_2) and the working set defined only with the equality constraint x_1+x_2=0, we have that g_k=(3 9) and that the system of equations is the following one:

$$\begin{array}{ccccccc} x_1 & + & 3x_2 & + & \lambda & = & -1 \\ 3x_1 & + & 9x_2 & + & \lambda & = & -3 \\ x_1 & + & x_2 & & & = & 0 \end{array}$$

This system has a single solution: x_1=1.5, x_2=-1.5, λ=0. d=(1.5, -1.5) corresponds to the direction to move from p=(1/2,1/2). Due to the constraints, we arrive to p=(1,0) that is the single solution of the problem. □

Putting all the results together, we have that linear independency in H leads to single solution minimization problems, and that linearly dependency can lead to multiple solutions but also to a single solution.

3.3 Dealing with multiple solutions when dependent columns in matrix H.

Due to the possibility of having minimization problem with multiple solutions, and due to the fact that not all solutions are equally good, we introduce in this section a procedure to select the solution in such case.

As it has been said, when several weighting vectors are possible, it is better not to accumulate the weights into a single source but to distribute them among the sources (maximum dispersion on the values). Entropy has often been used (see e.g. [Yager, 1993, O'Hagan, 1988, Carbonell et al., 1997]) to measure the dispersion of weighting vectors. This is,

$$E(p) = -\sum_{i} p_i \ln p_i$$

To compute this expression, it is convenient to extend the domain so that zero values (zero weights) are allowed. This is, an expression which appears formally as "0 ln 0" is defined to be zero [Ash et al., 1965].

The entropy is defined so that maximal entropy (with the restriction that $\sum p_i = 1$) corresponds to maximal dispersion and is achieved when all the weights are defined as $p_i = 1/N$. Thus, with a maximum entropy, the influence of a particular information source is minimized. According to this, if we have that there exist several solutions of the quadratic problem stated above with the same minimal distance Δ, we want to select the one with maximal entropy.

Due to the fact that a priori we do not know which is the minimal distance Δ, we need to solve the general problem in two steps. In the first step, we solve the minimization problem stated above, and determine if there is a single solution or there exist several ones and settle the minimal distance Δ. In the second step, we find the weighting vector p that maximizes the entropy and has a minimal distance Δ. The second step is a non-linear optimization problem with non-linear constraints. This algorithm is described below:

```
Procedure Learning_Weights_WM_2Steps
                                  (p: weighting vector,
                                   Q: quadratic matix,
                                   c: vector) returns weighting vector is
begin
    Determine p and $\Delta$:=D(p) such that:
        Min D(p) = (0.5) p' Q p + c' p (2)
        such that
        $\sum_{i=1}^{N} p_i = 1$                                        (3)
        $p_i \geq 0$                                                     (4)
    if there exist more than a single p such that $\Delta$=D(p) then
        Determine p such that:
            Min - E(p) = $\sum_i p_i \ln p_i$
            such that
            (0.5) p' Q p + c' p = $\Delta$
        $\sum_{i=1}^{N} p_i = 1$
        $p_i \geq 0$
    end if;
    return p;
end procedure;
```

3.4 Learning the weights for the OWA operators

The learning of weights in the OWA operator is analogous to the case of the weighted mean. Recall that the OWA operator is just a weighted mean once the elements to weight are ordered according to their value. Therefore, we can apply the method described above to learn the weights in the OWA. The only difference is the definition of the distance D(p). In this case, we have that:

$$D(w) = \sum_{j=1}^{M} \left(\sum_{i=1}^{N} OWA_W(a) - m^j \right)^2$$

that, according to the definition of the OWA operator, corresponds to:

$$D(w) = \sum_{j=1}^{M} \left(\sum_{i=1}^{N} a^j_{\sigma(i)} w_i - m^j \right)^2$$

with σ being a permutation as in the definition of the OWA operator in Section 2.

3.4.1. Learning the weights in the WOWA operator

Similar approach can be applied to the WOWA operator. In this case, however, the distance has to be defined in terms of the operator and with equality restrictions with the two sets of weights. Thus, the general form of the method is as follows:

$$\text{Min } D\ (p,w) = \sum_{j=1}^{M} \left(\sum_{i=1}^{N} WOWA_{w,p}(a^j) - m^j \right)^2$$

such that

$$\sum_{i=1}^{N} p_i = 1$$
$$\sum_{i=1}^{N} w_i = 1$$
$$p_i \geq 0,\ w_i \geq 0$$

The problem formulated in this way can also be solved by means of optimization algorithms. However, in this case the minimization problem is not a quadratic problem (due to the computation of the WOWA and the weights w_i). Therefore the active set method described in Section 3 is not adequate and more complex software is needed. This software needs to compute a gradient vector and the Hessian matrix. Both elements can be built from the definition of the WOWA operator and the interpolation method defined to build the quantifier from the weights w_i.

4 Learning some weights

We consider in this Section some examples of the application of the weighting vector learning. We begin with two small examples. One of them taken from the literature [Filev et al., 1998] that learns the weights for an OWA operator. We show that the method introduced here is faster and give better results than the one introduced in that work. This section ends with larger examples where the dimension of the weighted mean is 4 and 8 respectively and where the number of rows in the matrix is 150 and 4177.

4.1 Toy examples

Example 2. The example consists on a data matrix H (with the values of the information sources) with 10 examples and 5 weights, and its corresponding solution vector d. With a single data matrix H, we consider three output vectors d_1, d_2 and d_3. The first one has been computed using a known weighting vector p=(0.1, 0.2, 0.3, 0.4, 0.0). The vectors d_2 and d_3 are perturbations from d_1. The perturbations of d_3 are greater than those of the d_2. Table 2 displays the data matrix and the three output vectors. Each row corresponds to an example and in the last column we have its respective output.

0.3	0.4	0.5	0.1	0.2	0.3	0.32	0.2
0.2	0.1	0.4	0.1	0.5	0.2	0.25	0.1
0.2	0.5	0.8	0.0	0.1	0.36	0.37	0.2
1.0	0.5	0.3	0.6	0.7	0.53	0.6	0.4
0.2	0.1	0.1	0.1	0.7	0.7	0.76	0.5
0.4	0.8	0.2	0.8	0.6	0.58	0.64	0.3
0.3	0.2	0.1	0.4	0.3	0.26	0.29	0.1
0.6	0.8	0.7	0.2	0.5	0.51	0.56	0.3
0.1	0.5	0.2	0.6	0.4	0.41	0.45	0.3

Table 2. Data matrix H and three solution vector d_1, d_2 and d_3

The weighting vectors p_1, p_2 and p_3 and the corresponding minimal distances $D(p_i)$ that are solutions of the minimization problems resulting from H and the three vectors di are:

$p_1 = (0.100, 0.200, 0.300, 0.400, 0.000)$ $D(p_1) = 0.0$
$p_2 = (0.080, 0.350, 0.153, 0.218, 0.198)$ $D(p_2) = 0.008686595012843412$
$p_3 = (0.058, 0.000, 0.420, 0.522, 0.000)$ $D(p_3) = 0.2078640861200184$

As expected, the weights in the first example are the ones used to generate the solution vector. In the second and third column, as larger perturbations were performed, the resulting weighting vector is less and less similar than the first one. Also, D(p_i) increases in the last examples. In relation to computational issues, the first two results were obtained with the algorithm in a single step, while in the third case three steps were needed. In all three cases, the initial weighting vector was p=(.2, .2, .2, .2, .2). In the last case, to find a feasible solution two constraints have to be added in the working set. These relate to the weights p_2 and p_5. This is due to the fact that the global minimum without equality constraints would be obtained with negative weights. □

In [Filev et al., 1998], Filev and Yager presents an alternative method for learning the weights in the OWA operator. It is based on the use of the gradient technique. The followig example shows the results obtained with our algorithm based on active sets. We show that it is more efficient and that more exact results can be obtained with it.

Example 3. Let N be the set of examples taken from [Filev et al., 1998] and given in Table 3. Each sample consists of 4 information sources and the corresponding aggregated value. In order to apply the previous learning algorithm, as the weights in the OWA have to be considered in relation to the position of the elements, we need to order the elements. After ordering them we get the matrix in Table 4.

0.4	0.1	0.3	0.8	0.24
0.1	0.7	0.4	0.1	0.16
1.0	0.0	0.3	0.5	0.15
0.2	0.2	0.1	0.4	0.17
0.6	0.3	0.2	0.1	0.18

Table 3. Data matrix H and solution vector d (taken from [Filev et al., 1998])

0.8	0.4	0.3	0.1	0.24
0.7	0.4	0.1	0.1	0.16
1.0	0.5	0.3	0.0	0.15
0.4	0.2	0.2	0.1	0.17
0.6	0.3	0.2	0.1	0.18

Table 4. Data matrix after having ordered the values in each row

Applying the algorithm we obtain the following weighting vector:

w = (0.1031, 0.0, 0.2293, 0.6676)

For an initial weighting vector w equal to (.25, .25, .25, .25), we obtain w solving two systems of equations. First, the initial system with only the equality constraint, and second, the system with the second weight fixed to zero. For the same example, and with the same initial weighting vector w, the following solution is given in [Filev et al., 1998] after 150 iterations:

w = (0.08, 0.11, 0.14, 0.67)

The distance D(w) computed for both examples show that they are not equally good.

[Filev et al.,1998]: 0.002156

Our method: 0.001256

It is easy to see that our approach, besides of getting a solution in less iterations, is more precise in reproducing the aggregated value. □

4.2 Larger data sets

We have applied the procedure introduced in this paper to learn the weights to two large problems. As there are not existing databases for aggregation operators determination, we have selected two databases in the UCI Repository adapting them so that they are suitable for our purposes. We consider each row in the database as an example to learn. The values of the attributes are the inputs of the aggregation operator and the conclusion in the database is the output of the operator. To apply the algorithm described in this work, we have normalized all attributes into the unit interval. This is required because as it is well known, all aggregation procedures $\mathbb{C}$, and in particular the weighted mean, usually satisfy the following inequality:

$$\min (a_1, ..., a_n) = \mathbb{C} (a_1, ..., a_n) = \max (a_1, ..., a_n)$$

This equation is not always fulfilled if each value a_i is in a completely different scale. Putting them all in a single scale we can have a better approximation of this equation (although not always satisfied).

Besides of the scalation, we have modified the database so that only numerical attributes were considered. We have replaced categorical scales by numerical ones.

Example 4. Let the set of examples be the ones corresponding to the Iris database. In this case, examples are defined by means of the four attributes displayed in Table 5. To apply our procedure we have rescaled the value x for each attribute in the following way: $(x - x_{min}) / (x_{max} - x_{min})$, where x_{min} and x_{max} are the minimal and maximal values of the corresponding attribute.

The database consisting on the measurements for 150 iris of three classes (iris-setosa, iris-versicolor, iris-virginica) is intended to be used to determine the class of iris once the measurements for all atributes are supplied. In our case, we have replaced the classes (iris-setosa, iris-versicolor, iris-virginica) by numerical values (1.0, 2.0 and 3.0).

Attributes	Domains =[Mn, Mx]
Sepal length in cm	Mn:4.3 Mx:7.9
Sepal width in cm	Mn:2.0 Mx:4.4
Petal length in cm	Mn:1.0 Mx:6.9
Petal width in cm	Mn:0.1 Mx:2.5
Output class	Mn:1.0 Mx:3.0

Table 5. Attribute and domains in the Iris database.

Iteration	weighting vectors	D(p)
1	0.250 0.250 0.250 0.25	9.8962
2	0.000 0.052 0.459 0.489	3.8796
3	0.000 0.000 0.452 0.548	3.2152
4	0.000 0.000 0.219 0.781	3.1544

Table 6. Weights in consecutive iterations for the Iris example.

The results obtained by our algorithm are the ones given in Table 6. We have used an initial weighting vector p=(0.25, 0.25, 0.25, 0.25). The results show, what is already known from several studies (see, for example, the file iris.names in the UCI repository): that the output is highly correlated with the attributes "petal length" and "petal width". □

Example 5. Let the set of examples be the ones corresponding to the Abalone database. The database consists on the measurements of 4177 abalone and it is intended to determine the age of the abalone using 8 attributes. The basic information for the attributes is given in Table 7. To apply the algorithm we have replaced the values corresponding to the first attribute "Sex" that used three categories (M, F and I (infant)) by three numerical values (1.0, 2.0, 3.0). We have also rescaled the value x for each attribute using the minimal and maximal values of the attribute.

Attributes	Domains =[Mn, Mx]	
Sex	Mn:1.0	Mx:3.0
Length in mm	Mn:0.075	Mx:0.815
Diameter in mm	Mn:0.055	Mx:0.65
Height in mm	Mn:0.0	Mx:1.13
Whole weight in grams	Mn:0.0020	Mx:2.8255
Shucked weight in grams	Mn:0.0010	Mx:1.488
Viscera weights in grams	Mn:5.0E-4	Mx:0.76
Shell weight in grams	Mn:0.0015	Mx:1.005
Output class	Mn:1.0	Mx:29.0

Table 7. Attribute and domains in the Abalone database.

Iteration	weighting vectors	D(p)
1	0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125	58.9998
2	0.111 0.129 0.138 0.208 0.193 0.000 0.080 0.140	52.6290
3	0.085 0.134 0.180 0.267 0.000 0.000 0.038 0.296	44.4066
4	0.079 0.133 0.188 0.291 0.000 0.000 0.000 0.309	43.0417
5	0.026 0.000 0.364 0.494 0.000 0.000 0.000 0.116	37.6085
6	0.018 0.000 0.369 0.524 0.000 0.000 0.000 0.088	37.5189

Table 8. Weights in consecutive iterations for the Abalone example.

The weighting vector obtained by the algorithm is given in Table 8. It can be seen that four of the variables are settled to be zero by the algorithm. □

The results presented in this work have been obtained with an implementation of the algorithm Learning_weights_WM in Section 3 using Java. The solution of the system in (5) is obtained by means of an LR decomposition. With respect to execution times, the best approximation for the Iris example is obtained in a Sun in 1.32 seconds, while it takes 14.71 seconds in the Abalone example.

5 Conclusions

In this paper we have studied some aspects related with the learning of weights for some aggregation operators. In particular, we have considered the weighted mean, the OWA and the WOWA operator. We have described an algorithm based on active set methods, and we have applied them to two toy problems and to two larger ones to see the suitability of our approach. The numerical comparison with an existing method in the literature has shown that our method obtains a better solution in a few steps.

As a future work we plan to apply the procedure to other aggregation operators, and to study in more detail in which conditions the system has multiple solutions. In addition, we plan to change the procedure to compute the linear system (5) to deal better with numerical instability.

6 Acknowledgements

The author acknowledge partial support of the CYCIT project SMASH (TIC96-1138-C04-04). The author is also indebted to Jordi Castro and Lluís Godo for helpful comments and suggestions.

7 References

[1] Aczél, J., (1984), On weighted synthesis of judgements, *Aequationes Math.*, 27 288-307.

[2] Aczél, J., Alsina, C., (1986), On synthesis of Judgements, *Socio-Econom Plann. Sci.*

[3] Ash, R., (1965), Information theory, Interscience publishers.

[4] Carbonell, M., Mas, M., Mayor, G., (1997), On a class of Monotonic Extended OWA Operators, Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (IEEE-FUZZ'97), Barcelona, Catalunya, Spain, 1695-1699.

[5] Filev. D. P., Yager, R. R., (1998), On the issue of obtaining OWA operator weights, *Fuzzy Sets and Systems,* 94 157-169.

[6] Fodor, J., Marichal, J.-L., Roubens, M., (1995), Characterization of the Ordered Weighted Averaging Operators, *IEEE Trans. on Fuzzy Systems,* 3:2, 236-240.

[7] Fodor, J., Roubens, M., (1994), *Fuzzy Preference Modelling and Multicriteria Decision Support*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

[8] Gill, P.E., Murray, W., Wright, M. H., (1981), *Practical Optimization*, Academic Press.

[9] Grabisch, M., Nguyen, H.T., Walker, E.A., (1995), *Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference*, Kluwer Academic Publishers, Dordreht, The Netherlands.

[10] López, B., Álvarez, S., Millán, P., Puig, D., Riaño, D., Torra, V., (1994), Multistage vision system for road lane markings and obstacle detection, Proceedings of *Euriscon '94*, Malaga, pp. 489-497.

[11] López de Mántaras, R., Amat, J., Esteva, F., López, M., Sierra, C., (1997), Generation of unknown environment maps by cooperative Low-cost robots, *Proc. of the 1st Int. Conf. on Autonomous Agents*, Marina del Rey, CA, USA.

[12] Luenberger, D. G., (1973), *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Menlo Park, California.

[13] O'Hagan, M., (1988), Aggregating template or rule antecedents in real-time expert systems with fuzzy set logic, *Proceedings of the 22nd Annual IEEE Asilomar Conference on Signals, Systems and Computers,* Pacific Grove, CA, 681-689.

[14] Stoer, J., Bulirsch, R., (1980), Introduction to numerical analysis, Springer-Verlag.

[15] Torra, V., Cortés, U., (1995), Towards an automatic consensus generator tool: EGAC, *IEEE Transactions on Systems, Man and Cybernetics,* 25:5 888-894.

[16] Torra, V., (1997), The Weighted OWA operator, *International Journal of Intelligent Systems*, 12 153-166.

[17] Torra, V., (1999), The WOWA operator and the interpolation function W*: Chen and Otto's interpolation method revisited, *Fuzzy Sets and Systems* (in press).

[18] Torra, V., (1998), *On the integration of numerical information: from the arithmetic mean to fuzzy integrals,* Research Report DEI-RR-98-10, Computer Engineering Department, Universitat Rovira i Virgili.

[19] Yager, R.R., (1993), Families of OWA operators, *Fuzzy Sets and Systems*, 59 125-148.

[20] Yager, R.R., (1988), On ordered weighted averaging aggregation operators in multi-criteria decision making, *IEEE Trans. on SMC,* 18 183-190.