This is the authors final peer reviewed version of the item published as:


**Copyright**: 2003, Wiley Periodicals, Inc.
How to build aggregation operators from data?

G.Beliakov

School of Information Technology

Deakin University

225 Burwood Hwy, Burwood 3125, Australia

email: gley@deakin.edu.au

Abstract
This paper discusses a range of regression techniques specifically tailored to building aggregation operators from empirical data. These techniques identify optimal parameters of aggregation operators from various classes (triangular norms, uninorms, copulas, OWA, generalised means, compensatory and general aggregation operators), while allowing one to preserve specific properties, such as commutativity or associativity.

Keywords: aggregation operators, fuzzy sets, restricted least squares, regression splines

1. Introduction
Aggregation operators serve as a tool for combining various degrees of membership into one numerical value, and are routinely used in many applications of fuzzy set theory. The theoretical properties of more than 90 different families of operators have been extensively studied and the reader is referred to [17, 30, 32, 40] for an overview. Yet when it comes to choosing an operator for a particular application, there are few tools to help practitioners. Choosing an operator on the basis of theoretical properties is of little value, because usually these properties define a very large class of operators
rather than a particular formula. The notable exceptions are max, min and Lukasiewitz operators.

Zimmermann [40] lists several criteria based on which aggregation operators can be selected. Among them empirical fit is probably the most useful as it has a direct quantitative interpretation. In most cases the problem of choice of the operator is translated into some sort of regression problem, like least squares fit.

However it is important to realise that fitting aggregation operators to data requires specialised regression techniques due to essential theoretical and semantical properties of these operators. While in some cases these properties may be satisfied automatically, in other cases they need to be enforced, for example by defining constraints. This paper will examine in detail several specialised regression methods applicable to various classes of aggregation operators. The benefit of using special techniques is that they are tailored to this particular problem, and incorporate in one form or another the semantics of aggregation operators. These techniques also have computational advantages, including the speed and quality of the solution.

The problem of fitting the parameters of aggregation operators to empirical data was examined by several authors, including [41, 19, 25, 18]. The method of monotone regression splines was applied to this problem in [4, 9, 12], and approximation of additive generators was discussed in [5, 8, 11]. The present paper systematises various approaches and solution techniques, presents them in a unique consistent notation, and also generalises a few of them. A software package which implements the described methods is also available.

2. Problem formulation

Consider the problem of fitting an aggregation operator $f(x_1, x_2, ..., x_n)$ to the empirical data. The empirical data consists of a list of pairs/triples/n-tuples of membership values to be aggregated $\{(x_1^k, x_2^k, ..., x_n^k)\}_{k=1}^K$, and the corresponding compound membership values $\{d^k\}_{k=1}^K$, measured experimentally (e.g., through questionnaires [41]).
The aggregation operator is an $n$-place function $f : [0,1]^n \rightarrow [0,1]$, non-decreasing in all arguments and satisfying $f(0) = 0$, $f(1) = 1$. This function is also called general aggregation operator [30], and the above properties constitute the minimal set of properties aggregation operators must satisfy. We will also require continuity of $f$, as this property is important from practical point of view (as opposed to theoretical constructions [29]).

Additional properties define particular classes of aggregation operators. For example, commutativity, associativity and boundary condition $f(x,0)=0$ define the well known class of triangular norms.

We formulate the problem as follows.

\[
\text{Minimise } \|f(x_1, x_2, \ldots, x_n) - d\| \\
\text{Subject to } f \text{ belonging to a given class of aggregation operators. } d \text{ denotes the } K-\text{vector of measured compound membership values.}
\]

We note from the beginning that generally this is an approximation, and not an interpolation problem. That is, $f$ need not fit the empirical data exactly, and the data itself may not satisfy the properties required from the operator (e.g., we may have data points that violate commutativity condition, hence no commutative operator can fit the data exactly). It is understood that empirical data may contain some measurement errors (noise), and hence we look for an operator that approximates the data. The norm in the expression (1) is usually $l_2$ (least squares regression), but can also be $l_p$ or $l_\infty$ (max norm).

3. Parametric techniques

In this section, the algebraic form of the aggregation operator is fixed in advance. However, the operator contains one or more unknown parameters whose values need to be determined from the data.

\textit{a) Constrained linear regression: Compensatory operators}
The first technique of fitting aggregation operators to the data was presented in [41], where the authors took compensatory operators

\[ f(x_1, x_2) = (x_1 x_2)^w (1 - (1 - x_1)(1 - x_2))^{1-w} \]  

(2)
as their model. As the empirical data they took the students’ responses about the grades of membership of various objects in the fuzzy sets “container”, “metallic object” and then in the compound set “metallic container”. The authors compared the empirical data with the values given by the \( \text{min} \) operator (as the model for intersection) and established that \( \text{min} \) was not a good model in this situation. They took Eq.(2) and found the best value of the parameter \( w \) using ordinary least squares regression. The resulting compensatory operator significantly outperformed \( \text{min} \) with respect to the quality of data fit.

The authors of [41] further suggested the use of a linear combination of a conjunctive and disjunctive operators (usually dual to each other)

\[ f(x_1, x_2, \ldots) = wT(x_1, x_2, \ldots) + (1-w)C(x_1, x_2, \ldots) \]  

(3)
The best value of \( w \) in this model can be also found using linear regression.

Eqs. (2),(3) are very simple linear/loglinear models, which will be the starting point for our discussion. Mathematically, the regression problem is formulated as minimisation of the following expression for loglinear combination (2),

\[ \text{Minimise} \sum_{k=1}^{K} \left( T(x_1^k, x_2^k, \ldots, x_n^k)^w C(x_1^k, x_2^k, \ldots, x_n^k)^{1-w} - d^k \right)^2 \]  

(4)
s.t. \( 0 \leq w \leq 1 \), and

\[ \text{Minimise} \sum_{k=1}^{K} \left( wT(x_1^k, x_2^k, \ldots, x_n^k) + (1-w)C(x_1^k, x_2^k, \ldots, x_n^k) - d^k \right)^2 \]  

(5)
s.t. \( 0 \leq w \leq 1 \),

for linear combination (3). \( T(x_1, x_2, \ldots, x_n) \) and \( C(x_1, x_2, \ldots, x_n) \) represent any two given aggregation operators (such as a triangular norm and conorm). The best value of \( w \) in the least squares sense is the solution to the above minimisation problems.
The solution to problem (5) can be found explicitly by differentiating (5) with respect to \( w \) and equalling the derivative to 0:

\[
\sum_{k=1}^{K} \left( w(T(x_1^k, \ldots, x_n^k) - C(x_1^k, \ldots, x_n^k)) + C(x_1^k, \ldots, x_n^k) - d^k \right) \left( T(x_1^k, \ldots, x_n^k) - C(x_1^k, \ldots, x_n^k) \right) = 0
\]

or

\[
w = \frac{\sum_{k=1}^{K} \left( T(x_1^k, \ldots, x_n^k) - C(x_1^k, \ldots, x_n^k) \right) \left( d^k - C(x_1^k, \ldots, x_n^k) \right)}{\sum_{k=1}^{K} \left( T(x_1^k, \ldots, x_n^k) - C(x_1^k, \ldots, x_n^k) \right)^2}
\]

(6)

If the resulting value of \( w \) does not satisfy \( 0 \leq w \leq 1 \), it is chosen as either 0 or 1, whichever gives a smaller value of (5).

For the compensatory operator in the form

\[ f(x_1, x_2, \ldots, x_n) = T(x_1, x_2, \ldots, x_n)^w C(x_1, x_2, \ldots, x_n)^{1-w} \]

(7)

the method of solution is similar, but the equation is first linearised by taking logarithms:

\[ \ln f(x_1, x_2, \ldots, x_n) = w \ln T(x_1, x_2, \ldots, x_n) + (1-w) \ln C(x_1, x_2, \ldots, x_n) \]

Then the solution is expressed as Eq. (6), with \( T(), C() \) and \( d^k \) replaced with their respective logarithms.

There are two ways to extend model (5) for other operators. The first way is to increase the number of parameters. For instance, we may consider the following form of aggregation operators (see [13]):

\[
f(x_1, \ldots, x_n) = w_1 T_1(x_1, \ldots, x_n) + w_2 T_2(x_1, \ldots, x_n) + \ldots + w_J T_J(x_1, \ldots, x_n),
\]

(8)

\[ \sum_{j=1}^{J} w_j = 1 \]

The vector \( \mathbf{w} \) will denote the parameters that are fitted to the data. The mathematical formulation of the problem is similar to (5):

\[
\text{Minimise} \quad \sum_{k=1}^{K} \left( \sum_{j=1}^{J} w_j T_j(x_1^k, \ldots, x_n^k) - d^k \right)^2
\]

(9)
\[ \sum_{j=1}^{L} w_j = 1 \]

Notice, however, that because of the constraints on the components of vector \( w \), solution to problem (9) is significantly more complicated than that of problem (5). For problem (9) the domain of \( w \) is the unit simplex, and we cannot simply calculate and compare the values of (9) at all points on the boundary. Still the problem (9) is a linear least squares problem, and there are methods of its solution that take advantage of this property. Problem (9) is called the constrained linear least squares problem, and its solution is discussed in section 3. If instead of linear combination, functions \( T_j() \) are multiplied like in (7), one has to perform linearization by taking logarithms.

b) Nonlinear regression: fixed algebraic form

The second way to generalise problem (5) is to consider nonlinear dependence on the parameter \( w \). An example would be to find the operator from a given family (say, of triangular norms) that fits the data best. For instance, consider the Yager family of t-norms

\[
 f(x_1, x_2) = 1 - \min(1 - x_1)^w + (1 - x_2)^{1/w}, w > 0
\]

(10)

The task is to find the value of \( w \) so that (10) fits the data best. This can be formulated as minimisation problem

\[
 \text{Minimise} \quad \sum_{k=1}^{K} \left[ 1 - \min(1 - x_1^k)^w + (1 - x_2^k)^{1/w} - d^k \right]^2
\]

s.t. \( w > 0 \).

Because of nonlinearity of (10), solution of (11) is again more complicated than that of (5). The biggest challenge is that problem (11) will generally have many local minima, whereas the best value of \( w \) corresponds to the global minimum of (11). Provided that (11) depends on \( w \) sufficiently well ((11) satisfies Lipschitz condition), this global optimisation problem can be solved using a variety of methods (e.g., multistart local optimisation, simulated annealing, Lipschitz optimisation [26, 27]).
We can also consider operator families with several parameters, such as combinations of several parameterised families of triangular norms and conorms with unknown parameters. In this case the problem is formulated similarly to (11), with $\mathbf{w}$ now being a vector. Multidimensional global optimisation is a notoriously difficult problem (this is an instance of NP-hard problem), and can only be practically solved with a small number of components of $\mathbf{w}$. Recent developments on this subject are in [26, 34]. The new Cutting Angle method (CAM) [35, 3] can be efficiently applied to problems with 3-5 variables.

We will not pursue this line beyond noticing that if required, a few nonlinear parameters can be fitted to the data quite effectively using global optimisation techniques. Our reason is that one can hardly justify a specific algebraic form of the aggregation operator in the language of the original problem, let alone interpret several nonlinear parameters. For example, what would be advantages of one algebraic form over another, if both share exactly the same semantical properties (such as commutativity, or associativity)? In the following sections we will show how to build operators using these properties directly, without recurring to any specific algebraic form.

On the other hand, we will point out that $l_2$ norm in problems (4),(5),(9) (i.e., least squares fit) can be replaced with any other norm, such as $l_1$ or max norms, without drastic consequences for the algorithms. $l_1$ norm is useful for filtering out outliers in the data (so called robust regression), whereas max norm ensures that all data points are fitted accurately (uniform approximation). Due to developments in non-smooth optimisation (e.g., CAM, or discrete gradient method (DG) [2, 35]) there are few requirements on the objective function in (4),(5), (9) (e.g., it needs not be differentiable), and thus replacing the norm will be transparent for the user.

Finally, we reiterate that rarely data can be fitted by the operator exactly. Even though this may be an indication of inadequate choice of the operator family, in general data contain measurement errors and uncertainties, and fitting it exactly is counterproductive. A good validation tool is the plot the predicted vs. observed values, as it allows one to detect systematic under- or overestimation: any such bias is worrying.
4. OWA operators

OWA operators (Ordered Weighted Aggregation) were introduced by Yager in [37]. An OWA operator of dimension \( n \) is a mapping \( f : [0,1]^n \rightarrow [0,1] \), that has an associated weighting vector \( w = (w_1, ..., w_n)^T \), such that

\[
\sum_i w_i = 1, \quad w_i \in [0,1]
\]

and where

\[
f(x_1, ..., x_n) = \sum_i w_i x_{k_i}
\]

The vector \( k = (k_1, ..., k_n)^T \) is such permutation of \( (1,2, ..., n)^T \) that \( x_{k_i} \) is the \( i \)-th largest element in \( (x_1, ..., x_n)^T \). The fundamental aspect of OWA operator is that a particular weight \( w_i \) is associated with a particular ordered position \( i \) of the arguments. OWA operators include \( \min \), \( \max \) and arithmetic mean for the appropriate choice of vector \( w \).

In [37] Yager introduced a measure to characterise the type of aggregation performed by OWA operators. He calls it the orness measure. It is defined as

\[
\text{orness}(w) = \frac{1}{n-1} \sum_{i=1}^n (n-i)w_i
\]

It can be shown that orness of \( \max \) operator is 1, orness of \( \min \) operator is 0 and orness of the arithmetic mean is 0.5. Orness of other OWA operators lies in the unit interval. The measure of orness is frequently used as an additional constraint when determining weights of the operator. For instance in [22, 33] the weights are obtained by minimising the entropy of OWA operator, subject to the given measure of orness. These methods do not use empirical data.

Consider the problem of determining the vector of weights \( w \) from observations. The expression (1) now takes the form
\[
\min \left\| \sum_{i=1}^{n} y_i^k w_i - d^k \right\|_2^2 = \min \left\{ \sum_{k=1}^{K} \left( \sum_{i=1}^{n} y_i^k w_i - d^k \right)^2 \right\},
\]
\quad s.t. \sum_i w_i = 1, \ w_i \geq 0,

where \( y_i^k \) denote the \( i \)-th largest element of \( (x_1^k, ..., x_n^k) \) for every fixed \( k \). Passing from the original observation data to ordered values \( y_i^k \) is a simple mathematical trick which makes the regression problem linear.

Because of the constraints in problem (11), its solution is not that simple as that of the traditional linear regression problem. There are three ways the constrains can be dealt with. Filev and Yager [19] propose a nonlinear change of variables to transform the domain of \( w \) from the unit simplex to unrestricted domain. Then they used a standard local minimisation algorithm to minimise the transformed (no longer quadratic) error function. The second way is to use penalty function approach, and add appropriate penalty for violating the restrictions to expression (15), which is subsequently minimised using standard descent algorithms. The third method is to solve the restricted linear least squares problem directly, taking advantage of the linearity of (13) and the constraints. We describe this approach in some detail here.

Problem (15) is known as linear non-negative least squares problem with equality constraints (NNLSE) [31]. In the generic form it is written as

\[
\text{Solve } Ew = e, \ Aw = d, \ Gw \geq g.
\]  

where \( E, A \) and \( G \) are matrices of the system of exact equations, system of equations satisfied in least squares sense and the system of inequality constraints respectively [24]. Theoretical treatment of this problem and details of some of the algorithms to solve it are presented in [31, 23, 24], and one of the algorithms is available from netlib [16] as Algorithm 587. It is numerically very efficient, and allows one to solve even the problems in which matrix \( A \) has deficient rank (e.g., when the number of data is less than \( n \)).
It is now a rather simple task to formulate problem (15) as (16) and to construct the necessary matrices. Clearly, \( G \) is the identity matrix in our case, and \( g=0 \). \( K \) rows of the rectangular matrix \( A \) are formed by data values: \( A_{ki} = y_i^k \) and \( d \) is the vector of \( d^k \). Matrix \( E \) of linear restrictions contains only one row: \( E=(1,1,\ldots,1) \), and \( e=1 \).

Further, it is also possible to impose an additional constraint on the measure of orness of the operator, say \( \text{orness}(w)=\zeta \). This constraint is easily incorporated into the algorithm, and forms the second row of matrix \( E \), according to expression (14).

The described method of solution of (15) via non-negative linear least squares problem performs better than the other two mentioned approaches in respect to speed and the quality of the solution. For instance, as a result of non-linear change of variables proposed in [19], the expression to be minimised has multiple local minima, and the descent algorithms may converge to any of them, not necessarily to the solution of (15). This is clearly illustrated by the fact that the solution produced by the method from [19] may be inferior to that of NNLSE [11]. The size of the vector \( w \) may also present a problem to local minimisation algorithms, whereas the NNLSE algorithm easily handles a few hundreds of variables.

5. General aggregation operators

General (or generalised) aggregation operators are functions \( f:[0,1]^n \rightarrow [0,1] \), non-decreasing in all arguments and satisfying \( f(0) = 0 \), \( f(1) = 1 \) [30]. We limit ourselves to continuous general aggregation operators as these are the ones most likely to be of practical use.

From the formulation of this problem, the algebraic form of the operator is unknown (and unrestricted). All we want to do is to approximate given observation data by a monotone continuous surface, passing through the two specified boundary points. In this section we will describe a well known nonparametric regression technique of polynomial splines, and show how to apply it to aggregation operators. Splines are versatile in approximating functions of any shape, their computation is numerically stable, and additional constraints can be easily incorporated into regression algorithm.
To approximate aggregation operators we will use tensor-product splines, which are given by the expression

$$ S(x_1, \ldots, x_n) = \sum_{j_1=-p}^{N_1} \ldots \sum_{j_n=-p}^{N_n} b_{j_1 \ldots j_n} T_{j_1 \ldots j_n}(x_1, \ldots, x_n). $$

(17)

Here $n$ is the dimension of the spline, $b_{j_1 \ldots j_n}$ are spline coefficients that need to be determined from the data, $p$ is the degree of the spline, $N_i + p + 1$ is the number of spline knots with respect to the $i$-th variable, and $T_{j_1 \ldots j_n}$ are tensor products of univariate basis functions

$$ T_{j_1 \ldots j_n}(x_1, \ldots, x_n) = \prod_{i=1}^{n} T_{j_i}(x_i). $$

(18)

The univariate basis functions $T_j(x)$, $j = -p, \ldots, N$ are chosen in such a way that the monotonicity of the spline is expressed by a simple condition of non-negativity of the coefficients. They are related to the traditional B-splines [15], and can be computed as their linear combinations. More details on the properties of functions $T_{j_1 \ldots j_n}$ and methods of their construction are given in [10, 4, 5, 12].

Now we can formulate the problem of approximating general aggregation operators as Eq.(1), which now takes the form

$$ \text{Minimise } \left[ \sum_{k=1}^{K} \left( \sum_{j_1=-p}^{N_1} \ldots \sum_{j_n=-p}^{N_n} b_{j_1 \ldots j_n} T_{j_1 \ldots j_n}(x_1^k, \ldots, x_n^k) - d^k \right)^2 \right] $$

(19)

s.t. $\sum_{j_1=-p}^{N_1} \ldots \sum_{j_n=-p}^{N_n} b_{j_1 \ldots j_n} \geq 0$, $j_i = -p + 1, \ldots, N_i$, $J_1 = -p, \ldots, N_1$, $i = 1, 2, \ldots, n$,

$$ S(0, 0, \ldots, 0) = \sum_{j_1=-p}^{N_1} \ldots \sum_{j_n=-p}^{N_n} b_{j_1 \ldots j_n} T_{j_1 \ldots j_n}(0, \ldots, 0) = 0, $$

and

$$ S(1, 1, \ldots, 1) = \sum_{j_1=-p}^{N_1} \ldots \sum_{j_n=-p}^{N_n} b_{j_1 \ldots j_n} T_{j_1 \ldots j_n}(1, \ldots, 1) = 1. $$

The inequality conditions in (19) express monotonicity of the spline with respect to each variable, and the equality constraints are the required boundary conditions.
Usually one does not require smoothness of the aggregation operator, hence we can take tensor product linear splines \((p=1)\) for simplicity reasons. Note however, that the resulting multidimensional spline is not piecewise linear.

At this point we observe, that problem \((19)\) is a variation of the NNLSE problem \((16)\). It is called LSEI problem (least squares with equality and inequality constraints), and is transformed to NNLSE by a range of methods [23]. The rows of matrix \(A\) in \((16)\) are formed by the values of basis functions \(T_{n_1/2\ldots n_m}\) at \(k\)-th observation point, inequality constraints in \((19)\) form matrix \(G\), which will be a tensor product of the identity and lower triangular matrices whose all non-zero elements are 1s, and matrix \(E\) will contain two rows, given by the values of basis functions at \(0\) and at \(1\) [10]. The number of columns in each matrix is \(\prod(N_i + p + 1)\).

For two-dimensional case the formulas in \((19)\) become simpler and can be easily implemented. The program code for the general case is available from the author. One should notice, however, that the number of coefficients in \((17)\) grows exponentially with the dimension of the operator \(n\). To determine these coefficients requires a huge number of observation data points. Hence this method is practical for small \(n\) (2-5). Further details on tensor product spline approximation are in [5, 8, 12].

6. Preservation of specific properties

In this section we extend the method of tensor product splines for general aggregation operators. The purpose is to incorporate several important semantical properties that may be required from the operator. Examples are commutativity, idempotency and boundary conditions \(f(x,0,0,\ldots,0) = x\) and \(f(x,1,1,\ldots,1) = x\).

a) boundary conditions

Boundary conditions of type \(f(x,0,0,\ldots,0) = x\) or \(f(x,1,1,\ldots,1) = x\) are associated with disjunctive and conjunctive behaviour respectively. They form part of the definition of triangular norms and conorms, and may be used in conjunction with commutativity condition. In some applications, these boundary conditions seem very natural.
When the aggregation operator is approximated with a tensor product linear spline, as in section 5, a way to ensure that one of these conditions is satisfied is to postulate

\[ S(t_j,0,0,...,0) = t_j \text{ or } S(t_j,1,1,...,1) = t_j , j = 0,...,N_i \]  

(20)

respectively, where \( t_j \) denote the knots of the spline (i.e., where the linear pieces are joined together). For \( n-1 \) variables being fixed, the tensor product spline \( S(x_1,...,x_n) \) is a continuous piecewise univariate linear function, hence conditions (20) at spline knots are sufficient to ensure the required boundary conditions.

If necessary, similar conditions are imposed for \( x \) at other positions in the argument list of \( f \) in the same way. In total, it may give at most \( \prod N_i \) equality constraints. These constraints are incorporated into the problem (19), and hence into (16), through the system of equations \( Ew = e \). Each constraint forms a separate row of matrix \( E \).

b) idempotency

Idempotency \( (f(x,x,...,x) = x) \) is another important semantical property that some aggregation operators may be required to satisfy. Essentially, it is imposed in the same way as boundary conditions, i.e., through setting values of the spline at certain points

\[ S(t_j,t_j,...,t_j) = t_j , j = 1,...,J . \]  

(21)

However, it is not sufficient to use only the knots of the spline (besides, the knots may not be on the diagonal of the unit hypercube), because \( G(x) = S(x,x,...,x) \) is not a piecewise linear function. Yet given the finite number of spline coefficients, \( J \) is a finite number. More precisely, for each \( n \)-dimensional rectangle (formed by the tensor product of one-dimensional intervals between the consecutive knots in each variable), which intersects the diagonal of the unit hypercube, we need at most \( n+1 \) interpolating conditions (21). Moreover, one can choose points \( t_j \) with relative freedom, and specify more interpolating conditions (21) than necessary, even more than the total number of spline coefficients. The system of equations \( Ew = e \) will still be consistent and the orthogonal factorisation of \( E \) will eliminate the redundant equations. The
reason is that the idempotency condition defines a linear subspace in the space of monotone tensor product splines.

c) commutativity

Commutativity implies that the approximating tensor product spline has to be defined only on the simplex $0 \leq x_n \leq x_{n-1} \leq \ldots \leq x_1 \leq 1$ rather than in the whole $[0,1]^n$. It reduces the number of coefficients to be determined by $n!$. Also, even though it is not critical, one should consider identical partitions of the unit interval with respect to each variable (spline knots). We consider two ways of imposing commutativity on tensor product spline, explicit and implicit.

Explicit method. In this approach one considers only the $1/n!$–th part of the matrix of spline coefficients, using its symmetry (e.g., in the case of two variables, one needs to know only half of the symmetric matrix of coefficients). The Eq. (17) takes the form

$$S(x_1,\ldots,x_n) = \sum_{j_1=1}^{N_1} \sum_{j_2=1}^{N_2} \ldots \sum_{j_n=1}^{N_n} b_{j_1\ldots j_n} T_{j_1\ldots j_n}(y_1,\ldots,y_n)$$

where $y_i$ denote the $i$-th largest element of $(x_1,\ldots,x_n)$. The least squares problem (19) is modified accordingly, and the coefficients are found through the solution of (16).

Implicit method. Instead of reducing the number of coefficients and basis functions, and modifying the least squares algorithm, one can opt to use the general approach (19) and modify the data instead. The original dataset $\{(x_1^k, x_2^k, \ldots, x_n^k), d^k\}_{k=1}^K$ is augmented by creating artificial data points $\{(x_{p_1}^k, x_{p_2}^k, \ldots, x_{p_n}^k), d^k\}_{k=1}^{Kn}$, where the indices $(p_1, p_2, \ldots, p_n)$ are all possible permutations of $(1,2,\ldots,n)$. This way one symmetrises the data, so that the approximating spline is also symmetric. For instance, in two-dimensional case one takes the original dataset $\{(x_1^k, x_2^k), d^k\}_{k=1}^K$ and augments it with $\{(x_2^k, x_1^k), d^k\}_{k=1}^K$. Then the general algorithm for problem (19) is applied to the compound dataset. The implicit method can be applied only to small $n$, because of quickly growing number of artificial data.
7. Associative operators: triangular norms

Associativity property of some families of aggregation operators is handy when extending a two argument function to the $n$-argument one, and also when the number of arguments is not fixed in advance. In the context of approximation, it drastically reduces the required number of experimental points: regardless of the dimension of the data, one effectively approximates a two argument function.

Important families of aggregation operators possess associativity, namely triangular norms, conorms and uninorms. However, associativity does not have a simple geometrical interpretation [36], which would allow one to impose geometrical restrictions like in the previous section. Therefore, if one wishes to restrict the class of aggregation operators built from the data to associative operators, one has to use an alternative technique.

One such technique is to use additive generators many of the associative operators possess. In this section we consider continuous Archimedean triangular norms (t-norms) and conorms. These operators have an associated monotone univariate function $g(x) : [0,1] \to [0,\infty]$, called the additive generator, such that

$$f(x_1, x_2) = g^{-1}(g(x_1) + g(x_2)).$$

(22)

$g^{-1}(x)$ denotes the pseudoinverse function [29].

Non-increasing additive generators represent triangular norms and non-decreasing generators represent triangular conorms. The additive generators are defined up to a positive multiplier. If $g(0) < \infty$ the triangular norm is nilpotent, and for $g(0) = \infty$ it is strict ($g(1) < \infty$ and $g(1) = \infty$ for t-conorms respectively). Detailed information about triangular norms and their characterisation is collected in the book [29].

The approach we use to impose associativity is to reconstruct from the data the additive generator, representative of a given triangular norm or conorm. Then the operator itself can be found from (22).

Write Eq.(22) as
Let us now use monotone univariate splines to approximate the unknown function \( g \):

\[
g(x) \approx S(x) = \sum_{j=p}^{N} b_j T_j(x)
\]

(24)

where basis functions \( T_j(x) \) are the same as in section 5. The monotonicity of the spline (24) is ensured by restricting coefficients \( b \) to non-negative (non-decreasing spline) or to non-positive (non-increasing spline). However, the empirical data is not given as the measured values of the function \( g \), but as \( \{(x_1^k, x_2^k, d^k)\}_{k=1}^{K} \). Hence we do not have the usual least squares approximation problem.

Eqs. (23) and (24) result in the following overdetermined linear system of equations

\[
\sum_{j=p}^{N} b_j T_j(x_1^k) + \sum_{j=p}^{N} b_j T_j(x_2^k) - \sum_{j=p}^{N} b_j T_j(d^k) = 0, \quad k=1,\ldots,K,
\]

and after factoring out \( b \),

\[
\sum_{j=p}^{N} b_j \left[T_j(x_1^k) + T_j(x_2^k) - T_j(d^k)\right] = \sum_{j=p}^{N} b_j \tilde{T}_j(x_1^k, x_2^k, d^k) = 0, \quad k=1,\ldots,K
\]

(25)

The system of \( K \) equations (25) must be solved in the least squares sense, like in (16), subject to restrictions on the coefficients \( b \), namely non-positivity (for t-norms), or non-negativity (for t-conorms). In addition, we need to impose the equality restrictions.

For all t-norms one restriction is common:

\[
\sum_{j=p}^{N} b_j T_j(1) = 0
\]

(26)

(and \( \sum_{j=p}^{N} b_j T_j(0) = 0 \) for t-conorms).

For nilpotent t-norms we have the second restriction
\[ \sum_{j=-p}^{N} b_j T_j(0) = 1 \quad \text{(27)} \]

(and \( \sum_{j=-p}^{N} b_j T_j(0) = 1 \) for t-conoms).

Since the additive generators are defined up to a positive multiplier, the choice of 1 on the right is arbitrary. Hence, we obtain the NNLSE problem (16) again, with the entries of the matrix \( A \) given by the values of functions \( \tilde{T}_j(x_1^k, x_2^k, d^k) \) at data points, the entries of matrix \( E \) given by values of \( T_j(0) \) and \( T_j(1) \) respectively, and the matrix \( D = -I \) (\( D=I \) for t-conoms).

For strict t-norms condition (27) must be changed to

\[ \sum_{j=-p}^{N} b_j T_j(0) = \infty \]

which would be impossible to achieve in a numerical algorithm. Replacing \( \infty \) with just a big number is useless: since t-norms are defined up to a multiplier, that would be equivalent to condition (27). To properly specify conditions on strict t-norms/conorms, we need the following

**Proposition.** Let \( \mu \) denote the smallest number among all \( \{ x_1^k, x_2^k, d^k \}_{k=1}^{K} \), which is not zero. Then the behaviour of the additive generator \( g(x) \) of the strict t-norm \( T \) on the interval \( x \in (0, \mu) \) cannot be determined from the data.

Proof. Observe that no value of \( g(x) \) on this interval is required for Eq. (23).

Consider two additive generators \( g_1 \) and \( g_2 \) coinciding on \([\mu,1]\) but different on \((0,\mu)\). If \( g_1 \) satisfies (23) for all data points (either exactly or in the least squares sense), then so does \( g_2 \). Because of arbitrary choice of \( g_2 \) on \((0,\mu)\), the regression problem has infinitely many solutions (that all coincide on \([\mu,1]\)). \( \square \)

Consequently, the choice of the functional form for \( g(x) \) on \((0,\mu)\) is not important for our regression problem, provided that it satisfies the general requirements, such as
monotonicity, continuity and \( g(0) = \infty \). For instance, one can use “well-founded”
generators [28]

\[
g_\mu(x) = \begin{cases} 
g(x), & x > \mu \\
\frac{1}{x} + g(\mu) - \frac{1}{\mu}, & x \leq \mu
\end{cases}
\] (28)

Thus, for strict t-norms one replaces condition (27) with

\[
\sum_{j=-p}^{N} b_j T_j(\mu) = 1
\] (29)

and lets

\[
g_\mu(x) = \begin{cases} 
\sum_{j=-p}^{N} b_j T_j(x), & x > \mu \\
\frac{1}{x} + 1 - \frac{1}{\mu}, & x \leq \mu
\end{cases}
\]

For t-conorms the solution is similar. Extension of Eq. (25) for \( n \)-dimensional
empirical data is trivial.

\textit{Copulas}

In [36, 29], a characterisation of t-norms, which are simultaneously copulas, is given.

Copulas are functions defined on the unit square by the conditions

\[
C(x, y) + C(x^*, y^*) \geq C(x, y^*) + C(x^*, y), \text{ provided } x \leq x^*, y \leq y^*,
\]

\[
C(x,0) = C(0,x) = 0
\]

\[
C(x,1) = C(1,x) = x
\]

Since some important t-norms are copulas (e.g., product, Lukasiewitz; Dubois-Prade,
Frank families and others), it may be necessary to impose an additional restriction on
the t-norm to be a copula. The characterisation theorem (see [36, 29]), states that for a
continuous Archimedian t-norm to be a copula, its additive generator \( g \) must be a
convex function. Such copula is called Archimedian copula. One may speak of
nilpotent and strict copulas, as well as their duals.
For our purpose of reconstruction of Archimedian copulas from the data, we need to impose the additional restriction of convexity on the approximating spline (24). This condition readily translates into linear restrictions on coefficients b, namely that the sequence of \{b_j\} must be non-decreasing (along with non-positivity of b themselves).

These restrictions are easily incorporated into the problem (16) through additional entries of the matrix D.

8. Uninorms

Uninorms, introduced by Yager in [39], are a generalization of t-norms and t-conorms. These are also monotone, commutative and associative functions, but they possess a neutral element \(e \in [0,1] : U(e, x) = e\), which could be different from 0 or 1.

When \(e=0\) the uninorm becomes t-norm, and when \(e=1\) it becomes t-conorm. The structure of uninorms was completely described in [21].

Applicability of uninorms to modeling aggregation operators was discussed in [38], and many of their properties were studied in numerous publications (e.g., [21, 29, 38, 20]). Uninorms exhibit both disjunctive and conjunctive behaviour on different parts of the domain \([0,1]^n\). The domain can be scaled to arbitrary hypercube \([a,b]^n\), and some operators routinely used in expert systems happen to be uninorms (e.g., MYCIN and PROSPECTOR’s operators are uninorms on \([-1,1]\) [14]).

An important class of uninorms, called representable uninorms, possess additive generators

\[
g : [0,1] \rightarrow [-\infty, \infty], \quad g(e) = 0, \\
g(0) = -\infty, \quad g(1) = \infty,
\]

which define the uninorm via

\[
f(x, y) = g^{-1}(g(x) + g(y)).
\]

Examples or representable uninorms and their plots are provided in [29]. Extension to \(n\)-dimensional case is obvious.
The technique we use to approximate uninorms from empirical data is very similar to the one we used for t-norms/conorms. We concentrate on the additive generator $g$, which will be approximated using monotone non-decreasing least squares spline $S$ (24). Eq. (25) remains valid, however condition (26) takes the form $S(e) = 0$. Since uninorms are also defined up to an arbitrary positive multiplier, we need to specify one of the solutions by using a condition similar to (29).

If the neutral element $e$ is fixed, reconstruction of the uninorm from the data is practically the same as that of t-norm/conorm. However, if $e$ is unknown, it must also be found from the data, and this makes the regression problem much harder. With respect to $e$, this is a non-linear non-convex optimization problem, which may possess multiple local minima. This problem is similar to that of regression splines with free knots [15].

The regression problem is formulated as follows.

\[
\text{Minimise } \sum_{k=1}^{K} \left( \sum_{j=-p}^{N} b_j \hat{T}_j(x_1^k, x_2^k, d^k) \right)^2,
\]

s.t. $b \geq 0$, $\sum_{j=-p}^{N} b_j T_j(e) = 0$ and $\sum_{j=-p}^{N} b_j T_j(\mu) = -1$.

The unknowns are $e$ and $b$. One can minimise the objective function in (30) with respect to $e \in [0,1]$ using global optimisation techniques. This one-dimensional problem can be solved using Piyavski-Shubert algorithm of deterministic global optimization, described in [26, 27]. One can also use a more general Cutting Angle method (CAM), formulated in multidimensional case [3, 35]. At each iteration of these methods, one repeatedly solves (30) with a fixed $e$, until the algorithm converges. Computationally, this is significantly more expensive than approximating t-norms. A numerical example illustrating this technique is provided in [11].

9. Quasi-arithmetic means

Another class of aggregation operators that possess generator functions is the class of quasi-arithmetic means. They are defined from [1]
where \( g \) is continuous and strictly monotone. A particular case is quasi-linear averaging operators \((g(x)=x^p)\) [18]. Special cases of quasi-linear operators are arithmetic, geometric and harmonic means. Generalised means are functions

\[
f(x_1, x_2, ..., x_n) = g^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} g(x_i)\right),
\]

(31)

The case of generalised quasi-linear means

\[
f(x_1, x_2, ..., x_n) = \left(\sum_{i=1}^{n} c_i x_i^p\right)^{1/p}, \quad c_i \geq 0, \sum c_i = 1, \quad p \in \mathbb{R},
\]

(33)

is considered in detail in [18].

\(a)\) quasi-arithmetic means

Reconstruction of quasi-arithmetic means from empirical data involves a procedure, similar to reconstruction of t-norms/conorms. Eq. (31) is written as

\[
\frac{1}{n} \sum_{i=1}^{n} g(x_i) - g\left(\frac{1}{n} \sum_{i=1}^{n} g(x_i)\right) = 0.
\]

The following regression problem is solved

\[
\sum_{j=-p}^{N} b_j \left[\frac{1}{n} \sum_{i=1}^{n} T(x_i^k) - T_j(d^k)\right] = \sum_{j=-p}^{N} b_j \tilde{T}_j(x_1^k, x_2^k, ..., x_n^k, d^k) = 0,
\]

(34)

s.t. \(b \geq 0\).

The only difference to (25) is the presence of \(1/n\) factor. Similarly to t-norms, the generating function \(g\) is defined up to a positive multiplier, but also up to an additive constant (i.e., if \(g(x)\) is the generating function for \(f\), so is \(a g(x)+b\)). Hence, two additional interpolation conditions to fix the solution are needed, e.g., \(g(0)=0\) and \(g(1)=1\). The numerical solution is performed using LSEI method (eq. (16)).
b) quasi-linear means

In this case the algebraic form of the generating function is known, and the task is to find the parameter \( p \), which fits the data best. This is done by solving the optimisation problem

\[
\text{Minimise } \sum_{k=0}^{K} \left( \frac{\sum_{i=1}^{n} x_i^k}{n} \right)^{1/p} - d^k \right)^2.
\] (35)

This problem is nonlinear, however it possesses the unique minimum, which can be obtained using any descent algorithm (e.g., Newton’s method). To show this, we remind the result from [18], which the authors call “the main property”: For any given arguments \( \mathbf{x} \) and any fixed coefficients \( \mathbf{c} \), the function (33) is monotone increasing in \( p \), and it is strictly increasing for \( c_i > 0 \), except \( x_1 = x_2 = \ldots = x_n \) or \( f(p; x_1, \ldots, x_n) = 0 \) only. The authors show that (33) is continuously differentiable with respect to \( p \), \( p \neq 0 \), and that the derivative is positive, except the mentioned cases.

The consequence of this is that the derivative of (35), call it \( \Phi(p) \), is also strictly monotone increasing (indeed, the expression in the brackets is increasing, and the derivative of (35) is the sum of such expressions with positive factors, that are the derivatives of (33)). Hence the solution of \( \Phi(p) = 0 \), if it exists, is unique. Of course, we exclude the special case when \( x_1^k = x_2^k = \ldots = x_n^k \) for all \( k \), which is meaningless in terms of the regression problem. The case \( f(p; x_1^k, \ldots, x_n^k) = 0 \) for all \( k \) leads to a constant (35) and implies multiple solutions, that are quasi-linear means with \( p \leq 0 \).

Hence, to find the optimal value of \( p \), we solve (35) on \( -\infty \leq p \leq \infty \) using a descent algorithm, cautiously treating special cases \( p = -\infty \) \((\text{min operator})\), \( p = \infty \) \((\text{max operator})\), and \( p = 0 \) \((\text{geometric mean})\).
c) *generalised quasi-linear means*

The difference with the previous case is the vector of coefficients \( \mathbf{c} \), whose components are different from \( 1/n \). The optimisation problem (35) is modified accordingly. For a given vector \( \mathbf{c} \), the main property from [18] remains valid, and (35) possesses the unique minimum, which can be found by using a descent algorithm. The set of special cases is slightly broadened to include cases \( c_i = 0 \).

However, when the vector \( \mathbf{c} \) is not fixed *a priori*, but is determined simultaneously with \( p \), the problem takes another dimension. It is now written

\[
\text{Minimise } \sum_{k=1}^{K} \left( \left( \sum_{i=1}^{n} c_i (x_i^k)^p \right)^{1/p} - d^k \right)^2, \tag{36}
\]

s.t. \( c_j \geq 0, \sum c_i = 1, p \) unrestricted.

In [18] this case is treated by using a descent algorithm with respect to all variables. Since for a fixed \( p \), however, the problem of finding \( \mathbf{c} \) is linear, we have a mixed linear-nonlinear regression problem. The linear and nonlinear variables can be separated, and (36) can be written as

\[
\min_{\mathbf{c}} \min_p \sum_{k=1}^{K} \left( \left( \sum_{i=1}^{n} c_i (x_i^k)^p \right)^{1/p} - d^k \right)^2 \tag{37}
\]

subject to the above restrictions on \( \mathbf{c} \). Eq.(37) is minimised with respect to \( p \) at the outer level (nonlinear problem), and for a fixed \( p \) (inner linear constrained problem), \( \mathbf{c} \) is found using LSEI problem (16).

The extra difficulty, not mentioned in [18], is that now the problem (36) may not have the unique minimum, even excluding the special cases. See [6, 7] for a discussion of similar multiple minima problem in data clustering. Consequently, using a local descent algorithm to find \( p \) will lead to a local, not necessarily global minimum of (36), depending on the starting point. Since this is only one nonlinear variable, though, the global minimum can be located using methods of deterministic global optimisation (such as Piyavski-Shubert method [26, 27], or Cutting Angle method [3, 35]).
Now we consider fitting functions (32) to the data. The difference with the method in section a) is the vector of coefficients $\mathbf{c}$, which now has to be found from the data simultaneously with $\mathbf{b}$. The regression problem (34) takes the form

$$
\sum_{j=-p}^{N} b_j \left[ c_i \sum_{i=1}^{n} T(x_i^k) - T_j(d^k) \right] = \sum_{j=-p}^{N} b_j \hat{T}_j(x_1^k, x_2^k, ..., x_n^k, d^k; \mathbf{c}) = 0,
$$

(38)

s.t. $\mathbf{b} \geq 0$, $\mathbf{c} \geq 0$, $\sum c_i = 1$.

As was the case with problem (36), (38) is a linear constrained regression problem provided that either $\mathbf{b}$ or $\mathbf{c}$ is fixed, but it is a nonlinear problem with respect to all variables, and there could be multiple locally optimal solutions. We can separate the variables, and proceed as follows. Treat the problem (38) with respect to $\mathbf{c}$ as a global optimisation problem, and for each fixed $\mathbf{c}$, solve (38) with respect to $\mathbf{b}$ as a constrained linear regression problem (LSEI method (16)):

$$
\min_{\mathbf{c}} \min_{\mathbf{b}} \left\{ \sum_{j=-p}^{N} b_j \hat{T}_j(x_1^k, x_2^k, ..., x_n^k, d^k; \mathbf{c}) \right\}^2,
$$

(39)

subject to the above constraints.

The Cutting Angle method [3, 35] is suitable to solve the outer problem: the domain is the unit simplex, natural for CAM, and the number of variables $n$ is in practice not very high (CAM is effective for $n \leq 10$).

10. Conclusion

This paper treats an important practical issue of determining parameters of aggregation operators from the empirical data. Aggregation operators are rather special functions, and they require special regression techniques, tailored to their semantically important properties. We examined all major families of aggregation operators (triangular norms/conorms, uninorms, means, OWA, compensatory operators, general operators), and presented a range of regression techniques tailored to each family. They are summarised in Table 1.
### Table 1. Regression techniques for families of aggregation operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Approximation method</th>
</tr>
</thead>
<tbody>
<tr>
<td>General aggregation operator</td>
<td></td>
</tr>
<tr>
<td>- Commutative</td>
<td>Monotone tensor product splines</td>
</tr>
<tr>
<td>- Idempotent</td>
<td>- explicit: tensor product spline on simplex</td>
</tr>
<tr>
<td>- Boundary conditions</td>
<td>- implicit: symmetrise the data</td>
</tr>
<tr>
<td></td>
<td>- additional interpolation conditions on diagonal</td>
</tr>
<tr>
<td></td>
<td>- additional interpolation conditions</td>
</tr>
<tr>
<td>Associative operators</td>
<td>Approximation of additive generators $g(x)$ with monotone spline</td>
</tr>
<tr>
<td>- continuous Archimedean t-norms/conorms</td>
<td>- monotone univariate spline</td>
</tr>
<tr>
<td></td>
<td>- interpolation condition $g(0)=1$</td>
</tr>
<tr>
<td></td>
<td>- well-founded generators</td>
</tr>
<tr>
<td></td>
<td>- convexity condition on $g$</td>
</tr>
<tr>
<td>- representable uninorms</td>
<td>- monotone spline and global optimisation with respect to neutral element</td>
</tr>
<tr>
<td>Means</td>
<td>Approximate generating function $g(x)$ with monotone spline</td>
</tr>
<tr>
<td>- quasiarithmetic</td>
<td>- monotone univariate spline</td>
</tr>
<tr>
<td>- quasilinear</td>
<td>- local optimisation to find parameter $p$</td>
</tr>
<tr>
<td>- generalised quasilinear</td>
<td>- global optimisation to find $p$, NNLSE to find weights</td>
</tr>
<tr>
<td>- generalised quasiarithmetic</td>
<td>- monotone spline, global optimisation with respect to weights</td>
</tr>
<tr>
<td>Ordered Weighted Aggregation</td>
<td>Least squares with equality and inequality constraints (LSEI)</td>
</tr>
<tr>
<td>Compensatory</td>
<td>Linear/loglinear least squares, LSEI if more than two operators are combined</td>
</tr>
<tr>
<td>Operators given in explicit</td>
<td>Global optimisation with respect to unknown parameters</td>
</tr>
<tr>
<td>algebraic form (parametric</td>
<td></td>
</tr>
<tr>
<td>families)</td>
<td></td>
</tr>
</tbody>
</table>

Some of the presented techniques rely on given algebraic form of the aggregation operators and use empirical data to identify optimal values of the parameters. Other methods take semantically important properties of operators, such as commutativity and associativity, and use non-parametric approximation techniques. The semantical properties are then translated into restrictions on the coefficients of operator representation.

Three generic optimisation techniques are employed: constrained linear regression, local optimisation and deterministic global optimisation. These techniques are frequently used in combination, and are adapted to the specifics of the regression.
problem at hand. The methods described in this paper are implemented into a software module, called *Aggregation operator approximation tool*, which is freely available from the author (http://www.it.deakin.edu.au/~gleb/software.html). We hope these methods will enrich the arsenal of tools available to practitioners.

References


