ARTICLE

DC optimization for constructing discrete Sugeno integrals and learning nonadditive measures

G. Beliakov[a], M. Gagolewski[a,b,c] and S. James[a]

[a]School of Information Technology, Deakin University, Geelong, Australia; [b]Faculty of Mathematics and Information Science, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warsaw, Poland; [c]Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland

ARTICLE HISTORY
Compiled December 11, 2019

ABSTRACT
Defined solely by means of order-theoretic operations meet (min) and join (max), weighted lattice polynomial functions are particularly useful for modeling data on an ordinal scale. A special case, the discrete Sugeno integral, defined with respect to a nonadditive measure (a capacity), enables accounting for the interdependencies between input variables.

However until recently the problem of identifying the fuzzy measure values with respect to various objectives and requirements has not received a great deal of attention. By expressing the learning problem as the difference of convex functions, we are able to apply DC (difference of convex) optimization methods. Here we formulate one of the global optimization steps as a local linear programming problem and investigate the improvement under different conditions.

Abbreviations:
DC = Difference of Convex
LP = Linear Programming

KEYWORDS
Aggregation functions; nonadditive measures; Sugeno integral; capacities; DC optimization

1. Introduction

This paper addresses computational aspects of learning nonadditive measures, often called fuzzy measures, or capacities [1–4]. Capacities are normalized monotone set functions (a particular class of cooperative games) which allow one to model decision making problems in which the decision criteria are interacting (complementary or redundant). Combined with nonlinear integrals, such as the Choquet integral, Sugeno integral and others [5], capacity-based decision making methods have been applied in numerous studies, see, e.g., [6–10].

A key feature of capacities is that they are not necessarily additive (for brevity, we call them nonadditive). Hence, they extend the traditional probability measures, enabling one to efficiently represent the variety of cases of interaction among multiple

CONTACT G. Beliakov. Email: gleb@deakin.edu.au

decision criteria [2,6,11,12]. It is generally accepted that an additive capacity reflects independence among the decision criteria, whereas superadditivity (more generally, supermodularity) reflects complementarity and subadditivity (submodularity) reflects redundancy. The Shapley interaction index (see [1,13]), as well as some alternative indices [12,14–16], measures the degree of interaction within a group of criteria.

The price for such flexibility and power of expression comes in the form of computational complexity. For n decision criteria there are $2^n - 2$ parameters that identify a capacity (and hence all input interactions), which becomes prohibitively high (even for moderate n) from two perspectives: eliciting the capacity values and their interpretation. It is unreasonable to expect domain experts to specify all decision criteria interactions, and be able to interpret the given parameters. There are two complementary approaches here: (a) simplifications to the capacities, which include k-additivity, symmetry, k-maxitivity, k-interactivity [17,18], and (b) automatic learning of the capacities from empirical data [6,7,19–31]. In the latter approach the decision makers may provide their target evaluations of the decision alternatives, or such evaluations can be observed and recorded automatically, and an optimal capacity can be identified by using regression methods.

In this paper we study capacity identification from data by using regression techniques, while at the same time relying on certain capacity simplifications in order to keep the computational complexity under control. When the decision process is modelled by the Choquet integral, the capacity learning problem can be achieved through solving a convex quadratic or linear programming problem, subject to $O(n2^n)$ monotonicity constraints [2,3,6,7,19,28].

Here we use the Sugeno integral as the decision model, which is more suitable for criteria expressed on an ordinal scale [32,33], a scale on which some variables can be compared and ranked, but where the differences between sequential values are meaningless. There are multiple contexts in which this is the case, in particular when using linguistic variables [34–37], for example, “high”, “normal”,“slightly enlarged”, “higher than normal”, “unremarkable”, “significantly low”, and so on. Fuzzy systems provide a natural way of representing the above mentioned concepts as fuzzy sets, and then performing operations on these sets, such as aggregation, implication and logical deduction. The rules (of a decision support system, for example) can be formulated in terms of a set of linguistic variables in a way that resembles natural language, and unlike black-box methods (e.g., based on deep learning and convolutional neural networks) facilitate their understanding and eventual acceptance by the end users.

We mention some previous studies in this area, in particular [19–22,25,26,29,30,38], which fall into the framework of learning capacities from empirical data by using the Sugeno integral. Our own recent studies have examined learning symmetric fuzzy measures by fitting the Sugeno integral to numerical data based on an L_1 fitness function [39,40] as well as with respect to the maximum or median residual [41]. In [42] we examined the use of non-smooth optimization methods for capacity learning. The challenge here, compared to the Choquet integral, is the fact that the learning problem becomes a non-convex non-smooth optimization problem with a large number of linear constraints.

In this paper we address the computational challenges of capacity identification in the Sugeno integral framework by using DC (difference of convex functions) programming. We formulate the fitting criteria by using a piecewise linear non-convex objective function and find its suitable DC decomposition. We then apply and numerically compare various optimization methods in the DC programming, paying particular attention to efficiently dealing with multiple linear constraints. We also employ various capacity simplification strategies in order to reduce the number of variables and constraints.

The main contributions of this work are: (a) the formulation of ordinal regression models for learning Sugeno integrals from linguistic data, (b) formulation of the appropriate non-smooth and DC optimization problems, (c) suitable simplifications to the capacities and (d) comparison of suitable computational tools to solve the associated optimization problems.

The paper is organized as follows. Section 2 provides some preliminary definitions and notation, with the capacity learning problem presented in Section 3. In Section 4 we show how the problem can be formulated as a DC problem and develop a number of sub-formulations for different stages of the optimization. In Section 5 we detail how the methods described for symmetric capacities can be generalized to non-symmetric capacities and discuss various simplification strategies. Section 6 is devoted to numerical experiments and benchmarking of the proposed optimization methods. Section 7 concludes.

2. Preliminary definitions

Fix $n \geqslant 2$. Sugeno integrals are defined with respect to capacities (also called discrete fuzzy measures). In the sequel we assume that the capacity and all the inputs range over the unit interval. Nevertheless, generally, the Sugeno integral can be computed with respect to data expressed over any ordinal set.

Definition 2.1. Let $N = \{1, 2, \dots, n\}$. A capacity is a set function $\mu : 2^N \to [0, 1]$ which is:

- monotonic, i.e., $\mu(A) \leqslant \mu(B)$ if $A \subset B$,
- normalized, i.e., satisfies $\mu(\emptyset) = 0$ and $\mu(N) = 1$.

Definition 2.2. The Sugeno integral with respect to a capacity μ and an input vector $\mathbf{x} \in [0, 1]^n$ is given by

$$S_\mu(\mathbf{x}) = \bigvee_{A \subseteq N} \left((\min_{i \in A} x_i) \wedge \mu(A) \right), \tag{1}$$

where $\vee$ is the maximum and $\wedge$ is the minimum.

Formula (1) can be rewritten by assuming the inputs have been pre-ordered non-decreasingly in advance, $x_{(1)} \leqslant x_{(2)} \leqslant \cdots \leqslant x_{(n)}$. Defining a weighting vector $\mathbf{h}$, $h_i = \mu(\{\sigma(i), \sigma(i+1), \dots, \sigma(n)\})$, corresponding with the chain of nested subsets induced by the ordering permutation of $\mathbf{x}$, $\sigma : N \to N$ such that the i-th smallest input $x_{(i)} = x_{\sigma(i)}$, yields:

$$S_{\mathbf{h}}(\mathbf{x}) = \max_{i=1,\dots,n} \min\{x_{(i)}, h_i\}. \tag{2}$$

As capacities are defined by $O(2^n)$ values, the Shapley indices are often used to estimate the relative importance of each variable. For a given capacity μ, the Shapley value [13] for each index i is given by

$$\phi(i) = \sum_{A \subseteq N \setminus \{i\}} \frac{(n-|A|-1)!|A|!}{n!} [\mu(A \cup \{i\}) - \mu(A)]. \tag{3}$$

The Shapley value is the vector $\vec{\phi}(\mu) = (\phi(1), \dots, \phi(n))$.

The Shapley value has been extended to the Shapley interaction index for all subsets and not just singletons [17,43]. Another index characterizing mutual dependence among the inputs is the nonadditivity index from [15]. There are also other indices, like the bipartition interaction index [12] and the non-modularity index [16] which can be used as alternatives to the Shapley interaction index.

It is worth noting that when the capacity is symmetric, i.e., $|A| = |B|$ implies that $\mu(A) = \mu(B)$ (the size of a subset depends only on its cardinality), then $\mathbf{h}$ does not depend on the original ordering of inputs. We also have $h_1 = \mu(N) = 1 \geqslant h_2 \geqslant \dots \geqslant h_n \geqslant 0$. Thus, the parameter space is only $n-1$ dimensional.

For the sake of completeness, let us also recall the definition of the discrete Choquet integral.

Definition 2.3. The Choquet integral with respect to a capacity μ and input vector $\mathbf{x}$ is given by

$$C_\mu(\mathbf{x}) = \sum_{i=1}^{n} (x_{(i)} - x_{(i-1)}) h_i, \tag{4}$$

where $x_{(i)}$ and h_i are defined as above and $x_{(0)} = 0$ by convention.

3. Learning problem formulation

When eliciting a capacity from empirical data, the dataset usually consists of observed input vectors $\mathbf{x}^{(k)}$ and observed outputs $y^{(k)}$, $k = 1, \dots, K$. The aim in data fitting is to find the parameters of a function f such that f matches the unknown function g that generated the data. Fitting (or model learning) can be quantified using different objectives, such as minimizing the sum of (absolute, squared) differences between the predicted $f(\mathbf{x}^{(k)})$ and observed $y^{(k)}$ values. Once the objective is chosen and the constraints on the parameters are specified, an optimization problem is solved so as to deliver the optimal model f. In our case the parameters of the model are the values of the capacity, and the function f is the Sugeno integral.

The problem of fitting capacities to data has been considered using both Choquet and Sugeno integrals, for example [6,19,21,22,28–30,38]. In particular for numerical data, learning capacities in the Choquet integral setting in the least absolute deviation sense has been performed by using linear programming techniques [7,27]. However when f is the Sugeno integral, the learning problem becomes much more complicated because of the non-convex multiextremal objective [39,40]. Furthermore, ordinal regression [26,44–46] implies a different objective: it is not required to fit the target outputs but rather preserve their relative ordering.

A significant challenge is to enforce monotonicity of the capacity, which translates into monotonicity of the nonlinear integral and has an important interpretation in the decision making context: an increase in the value of any decision criterion should not result in the decrease of the overall evaluation of an alternative. The number of (linear) monotonicity constraints is $O(n2^n)$.

Simplification strategies are employed to reduce the number of degrees-of-freedom and constraints of a capacity. The k-additive capacities [17] restrict marginal criteria interactions to subsets of cardinality at most k. However, except for the special case of 2-additive capacities, this technique does not reduce the number of monotonicity constraints. Another approach is to use k-interactivity [18] in which the capacity values for subsets of cardinality larger than k are fixed so as to maximize the capacity entropy.

When the capacity is symmetric, that is, the value $\mu(A)$ depends only on the cardinality of A, both the Choquet and Sugeno integral simplify to the ordered weighted averaging (OWA) function and the ordered weighted maximum function (OWMax, see, e.g., [2]), respectively. There are $n-1$ non-negative variables here, the vector $\mathbf{h}$ with one component $h_1 = 1$ fixed. The n constraints required are of the form $h_i \geqslant h_{i+1}$, $i = 1, \ldots, n-1$ and $h_n \geqslant 0$. We will treat this case in more detail in this section as a sufficiently simple special case in order to illustrate the other strategies.

We assume that the data is provided in the form of linguistic variable labels, which are mapped to integers and then to the respective rationals from $[0, 1]$. In the context of ordinal regression we are not required to fit the values $y^{(k)}$. Instead we require that the order of the observed values is matched by the prediction of the model. Thus, if $y^{(j)} \leqslant y^{(k)}$, we require $S_\mu(\mathbf{x}^{(j)}) \leqslant S_\mu(\mathbf{x}^{(k)})$. As the data may contain inaccuracies or may not correspond to a Sugeno integral with respect to any capacity, we expect inconsistencies in the two orders. Our aim is to minimize the inconsistencies.

Consider the following problem

$$\begin{gathered}\text{minimize } F(\mu) = \sum_{j,k=1}^{K} 0 \vee (S_\mu(\mathbf{x}^{(j)}) - S_\mu(\mathbf{x}^{(k)})) \\ \text{for all } j,k \text{ such that } y^{(j)} \leqslant y^{(k)} \\ \text{s.t. } \mu \text{ is monotone and } \mu(\emptyset) = 0, \mu(N) = 1.\end{gathered} \tag{5}$$

Note that whenever for $y^{(j)} \leqslant y^{(k)}$ we have $S_\mu(\mathbf{x}^{(j)}) \leqslant S_\mu(\mathbf{x}^{(k)})$, the corresponding term in the objective is 0. Also note that there are, in general, $2^n - 2$ variables $\mu(A)$ and $O(n2^n)$ linear inequality constraints representing monotonicity.

The objective F is not a convex function. We present some methods of solution in the next section based on its DC decomposition.

4. Optimization approaches

The objective function F in the preceding formulation is a piecewise linear non-convex function. Such functions require methods of non-smooth optimization [47], as they are not differentiable (especially at the critical points). Furthermore, there are many linear constraints on the feasible domain, which require specialized methods or introducing suitable penalty terms. However, in [40] we argued that the structure of these objective functions can be exploited. To illustrate this, first we consider the problem of fitting a symmetric capacity, which we later extend to other types of capacities. In this particular case, the objective becomes:

$$\begin{aligned} \text{minimize } F(\mathbf{h}) = \sum_{j,k=1}^{K} 0 \vee \left(\max_{i=1,\dots,n} \min\{x_i^{(j)}, h_i\} - \max_{i=1,\dots,n} \min\{x_i^{(k)}, h_i\} \right) \\ \text{for all } j,k \text{ such that } y^{(j)} \leqslant y^{(k)} \\ \text{s.t. } 1 = h_1 \geqslant h_2 \geqslant \dots \geqslant h_n \geqslant 0, \end{aligned} \tag{6}$$

where, for brevity of notation, we now assume that each $\mathbf{x}^{(k)}$ has been increasingly ordered in advance. The objective is a piecewise linear function, and as such it is Lipschitz and also a DC function. The latter is quite important as it allows us to apply the methods of DC optimization [48,49] to numerically solve a challenging global optimization problem.

4.1. DC decomposition

We now provide a DC decomposition of the objective $F = G - H$, where G and H are both convex functions, that will facilitate the calculation of the subdifferentials and application of the DC algorithm. We shall use the following identities:

$$\max_{i=1,\dots,n} \{g_i - h_i\} = \max_{i=1,\dots,n} \{g_i + \sum_{j \neq i} h_j\} - \sum_{i=1,\dots,n} h_i,$$

$$\min\{g, h\} = 0 - \max\{-g, -h\}, \ \ g, h \text{ linear},$$

$$\max_{i=1,\dots,n} \min\{x_i, h_i\} = \max_{i=1,\dots,n} \sum_{j \neq i} \max\{-x_j, -h_j\} - \sum_{i=1,\dots,n} \max\{-x_i, -h_i\}.$$

Note that the right hand side in each formula is the difference of two convex functions, so applying them recursively, and using the fact that the sum of convex functions is convex, we get

$$\begin{aligned} F(\mathbf{h}) = \sum_{j,k} 0 \vee \Bigg[\left(\max_{i=1,\dots,n} \sum_{m \neq i} \max\{-x_m^{(j)}, -h_m\} + \sum_{i=1,\dots,n} \max\{-x_i^{(k)}, -h_i\} \right) \\ - \left(\sum_{i=1,\dots,n} \max\{-x_i^{(j)}, -h_i\} + \max_{i=1,\dots,n} \sum_{m \neq i} \max\{-x_m^{(k)}, -h_m\} \right) \Bigg]. \end{aligned} \tag{7}$$

Lastly, applying the formula $\max\{0, G - H\} = \max\{H, G\} - H$, we get the DC formulation

$$F(\mathbf{h}) = \sum_{j,k: y^{(j)} \leqslant y^{(k)}} \max\{H_{jk}(\mathbf{h}), G_{jk}(\mathbf{h})\} - \sum_{j,k: y^{(j)} \leqslant y^{(k)}} H_{jk}(\mathbf{h}), \tag{8}$$

where

$$\begin{aligned}
G_{jk}(\mathbf{h}) = \quad & \max_{i=1,\dots,n} \sum_{m\neq i} \max\{-x_m^{(j)}, -h_m\} + \sum_{i=1,\dots,n} \max\{-x_i^{(k)}, -h_i\}, \\
H_{jk}(\mathbf{h}) = \quad & \sum_{i=1,\dots,n} \max\{-x_i^{(j)}, -h_i\} + \max_{i=1,\dots,n} \sum_{m\neq i} \max\{-x_m^{(k)}, -h_m\}.
\end{aligned}$$

Note that $G_{jk}(\mathbf{h}) = H_{kj}(\mathbf{h})$. We can also write

$$\max_{i=1,\dots,n} \sum_{m\neq i} \max\{-x_m^{(j)}, -h_m\} = \sum_{i=1,\dots,n} \max\{-x_i^{(j)}, -h_i\} - \min_{i=1,\dots,n} \max\{-x_i^{(j)}, -h_i\},$$

keeping in mind that $\min\{-x, -y\} = -\max\{x, y\}$ and $\max\{-x, -y\} = -\min\{x, y\}$.

Finally, we can rewrite the above objective as:

$$F(\mathbf{h}) = G(\mathbf{h}) - H(\mathbf{h}) \tag{9}$$

with

$$\begin{aligned}
G(\mathbf{h}) \quad = \quad & \sum_{j=1}^{K-1} \sum_{k=j+1}^{K} \max\left\{ \bigvee_{i=1}^{n} \min\{x_i^{(j)}, h_i\}, \bigvee_{i=1}^{n} \min\{x_i^{(k)}, h_i\} \right\} \\
& - \sum_{j=1}^{K} (K-1) \sum_{i=1}^{n} \min\{x_i^{(j)}, h_i\} & (10) \\
H(\mathbf{h}) \quad = \quad & \sum_{j=1}^{K} (j-1) \bigvee_{i=1}^{n} \min\{x_i^{(\sigma(j))}, h_i\} \\
& - \sum_{j=1}^{K} (K-1) \sum_{i=1}^{n} \min\{x_i^{(j)}, h_i\}, & (11)
\end{aligned}$$

where σ is a permutation such that for all j it either holds $y^{(\sigma(j))} < y^{(\sigma(j+1))}$, or that both $y^{(\sigma(j))} = y^{(\sigma(j+1))}$ and $\bigvee_{i=1}^{n} \min\{x_i^{(\sigma(j))}, h_i\} \geqslant \bigvee_{i=1}^{n} \min\{x_i^{(\sigma(j+1))}, h_i\}$. The latter qualification is needed to deal with ties. However, if all observations in $\mathbf{y}$ are unique, σ is just the ordering permutation of the outputs and it might be computed once at the data pre-processing step.

4.2. DC algorithm

DC decomposition allows one to apply specialized optimization algorithms to locate the minima of non-convex multiextremal functions, which is a very challenging optimization problem. Recall that convex functions have unique global minima (although not necessarily a single minimizer), and that many convex optimization methods with proven convergence are available. The DC optimization approach for $F = G - H$ is an iterative method, where at each iteration one replaces the convex function H with a suitable linear expression L, and hence solves a convex optimization problem for $G - L$, outlined below, by some standard technique. The DC iterations converge to a local minimum of F.

Let us recall the DC Algorithm (DCA) [50], which consists in iterating two steps.

```
Algorithm DCA.
Input: starting point $\mathbf{h}_0$ and $i = 0$
While convergence criteria not met do:
  1. Compute $\mathbf{y} \in \partial H(\mathbf{h}_i)$.
  2. Solve $\mathbf{z} = \arg\min_{\mathbf{h}} G(\mathbf{h}) - \langle \mathbf{h}, \mathbf{y} \rangle$ for a fixed $\mathbf{y}$.
       Set $\mathbf{h}_i = \mathbf{z}$ and $i = i + 1$.
```

Typical convergence criteria include reaching the difference between two successive iterations below a threshold, a threshold on the value of F or a predefined number of iterations. Note that the optimization problem at Step 2 is convex, and hence is solvable by many standard methods. Calculation of an element of the subgradient of H, denoted ∂H, in Step 1 is straightforward in our case, e.g., we can simply take one-sided derivatives of the terms in H. Let us then focus on Step 2 of the DC algorithm now.

4.3. Inner step

In this paper, we are going to test the following three approaches in terms of the speed and quality of the convergence:

(1) minimizing $G - L$ using a linear programming solver,
(2) minimizing $G - L$ using a generic derivative-free optimizer,
(3) transforming the problem domain and utilizing a bound-constrained approach.

4.3.1. Linear programming-based approach

The function G is piecewise linear and convex, and so is the objective in the DCA. One approach is to rely on linear programming (LP), whereby we convert Step 2 into a linear programming problem. Let us introduce auxiliary variables $\mathbf{a} \in R^{Kn}$, $\mathbf{b} \in R^{K(K-1)/2}$ in order to convert the expression for G into a linear function

$$\begin{aligned} a_{ij} &= \min\{x_i^{(j)}, h_i\}, \\ b_{jk} &= \max\{\bigvee_{i=1}^{n} a_{ij}, \bigvee_{i=1}^{n} a_{ik}\}. \end{aligned}$$

Then the associated linear programming problem at Step 2 is:

$$\begin{aligned}
\text{minimize } P(\mathbf{h},\mathbf{a},\mathbf{b}) \quad &= \quad -\sum_{i=1}^{n} y_i h_i + \sum_{j,k,j<k} b_{jk} - \sum_{i=1}^{n}\sum_{j=1}^{K}(K-1)a_{ij} \\
\text{s.t.} \quad & \quad b_{jk} \geqslant a_{ij}, i=1,\dots,n,\ \forall j<k \qquad (12)\\
& \quad b_{jk} \geqslant a_{ik}, i=1,\dots,n,\ \forall j<k \\
& \quad a_{ij} \leqslant x_i^{(j)}, i=1,\dots,n,\ j=1,\dots,K \\
& \quad a_{ij} \leqslant h_i, i=1,\dots,n,\ j=1,\dots,K \\
& \quad h_i \geqslant h_{i+1},\ i=1,\dots,n-1 \\
& \quad h_1 = 1.
\end{aligned}$$

This problem is solved by the standard Simplex method. Note that the same set of constraints is involved irrespective of the value $\mathbf{y}$ obtained at Step 1 of the DCA. For this reason, as DCA iterates, we have a sequence of similar LP problems at Step 2, which only differ by some of the coefficients in the objective. Hence, from the practical implementation perspective, one can set up a generic LP model and update it from iteration to iteration by changing the coefficients in the objective, rather than setting a new LP problem every time.

4.3.2. Derivative free optimization-based approach

An alternative to the LP formulation is to use a nonlinear derivative free optimization method, accounting for the linear constraints on $\mathbf{h}$. Here we employ Powell's constrained derivative free method LINCOA [51], or an alternative method, such as the truncated codifferential method [52], where we add piecewise linear penalty terms to the objective for constraint violations. We compare both approaches numerically in Section 6.

4.3.3. Change of variables and bound-constrained approach

We can also consider a change of variables and instead of optimizing w.r.t. nonincreasing $\mathbf{h} \in [0,1]^n$, consider the domain consisting of $\boldsymbol{\delta} = (\delta_1,\dots,\delta_{n-1})$ with box-constraints only $0 \leqslant \delta_i \leqslant 1$, $i=1,\dots,n-1$. Box constraints are convenient as there are many optimization methods tailored to dealing with them. A given $\boldsymbol{\delta}$ translates to the original $\mathbf{h}$ by means of the (reversed) cumulative sum:

$$h_i = \sum_{l=1}^{n-i+1} \delta_l$$

for $i = 2,\dots,n$. Moreover, $h_1 = 1$, which can be eliminated from the minimization problem, leaving the constraints $h_i = \sum_{l=1}^{n-i+1} \delta_i \leqslant 1$ for all $i = 2,\dots,n$. At the moment, only the constraint $h_2 \leqslant 1$ is non-redundant, but we next show that these constraints can all be eliminated as unnecessary.

Note that as the data belongs to the unit interval, it holds that $\min\{x_i^{(j)}, h_i\} = \min\{x_i^{(j)}, \min\{1, h_i\}\}$, as $x_i^{(j)} \leqslant 1$. Therefore the values of the terms in (9) do not change if we allow $h_i > 1$. Hence if an optimal value for the objective is attained at $h_i = h_i^*$, then it is also attained at all $h_i > h_i^*$.

We can rewrite the objective in terms of the new variables $\boldsymbol{\delta}$ as $\tilde{F}(\boldsymbol{\delta}) = \tilde{G}(\boldsymbol{\delta}) - \tilde{H}(\boldsymbol{\delta})$, which satisfy only non-negativity constraints. The variables h_i, $i = 2, \dots, n$ can be obtained through matrix-vector multiplication $[h_2, \dots, h_n] = \mathbf{A}\boldsymbol{\delta}$, where $\mathbf{A}$ is an invertible matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \iddots & \\ 1 & 1 & 1 & & \\ 1 & 1 & & & \\ 1 & & & & \end{bmatrix}, \qquad \mathbf{A}^{-1} = \begin{bmatrix} & & & & 1 \\ & & & 1 & -1 \\ & & \iddots & \iddots & \\ & 1 & -1 & & \\ 1 & -1 & & & \end{bmatrix}.$$

It is straightforward that the convexity and DC decomposition of F in (9) still hold under this linear change of variables. Furthermore the calculation of the subgradients of H is also done through (11), by using $\frac{\partial H}{\partial \boldsymbol{\delta}} = \mathbf{A}^T \frac{\partial H}{\partial \boldsymbol{h}}$, where the partial derivatives denote the column vectors of the (one-sided) gradients with respect to $\delta_1, \dots, \delta_{n-1}$ and $h_2, \dots, h_n$.

For this reason we can now use additional derivative-free methods such as BOBYQA [53] for bound-constrained minimization (directly applied to F or within the DC algorithm), or use DCA with the linear programming formulation (12) in terms of variables $\boldsymbol{\delta}$. See results and discussion in Section 6.

5. Non-symmetric capacities

We now address the issue of fitting general capacities, where the Sugeno integral is given by Eq. (1), and the values $\mu(A)$ satisfy monotonicity and boundary conditions $\mu(A \cup \{i\}) \geqslant \mu(A)$ for all $i \notin A$ and all $A, |A| < n$, and $\mu(N) = 1$. Here the objective (9) remains essentially the same, but we change the variables $\mathbf{h}$ to $\mu \in R^{2^n}$, and also change expressions of the form $\min\{x_i^{(j)}, h_i\}$ to $\min\{x_i^{(j)}, \mu(\{\sigma(i), \sigma(i+1), \dots, \sigma(n)\})\}$, where σ is a permutation such that $x_{\sigma(i)}^{(j)}$ is the i-th smallest element of vector $\mathbf{x}^{(j)}$.

Hence for general capacities we have a much larger number of variables and constraints. Nevertheless the DC decomposition in (9) remains. Similarly to the case of symmetric capacities, the DCA can be applied. The convex piecewise linear problem at Step 2 can be solved either by using a derivative free method such as LINCOA or the truncated codifferential method with a much larger number of penalty terms, or by using linear programming similar to formulation (12), with the corresponding increase of the number of variables and constraints. The latter could offer significant advantages because (a) handling a large number of linear constraints is efficient using LP methods, and (b) the system of constraints is very sparse, and LP software is well suited to handling sparse matrices.

Still, even moderate dimension n leads to an excessively large number of monotonicity constraints. As a way of reducing the complexity of capacities, the concept of k-order fuzzy measures was developed [17,54–56]. In the framework of the Choquet integral, k-additive capacities have gained popularity, while in the framework of the Sugeno integral, k-maxitive capacities are used [20,31].

The k-maxitive capacities, for a particular $1 \leqslant k \leqslant n$, satisfy the condition $\mu(A) = \max_{B \subset A} \mu(B)$ for $|A| > k$. This implies that for at least one A of cardinality k we have $\mu(A) = 1$. Fitting the Sugeno integral to ordinal data subject to k-maxitivity was recently performed in [31] using Simulated Annealing optimization.

It is possible to specify the objective (5) under the condition of k-maxitivity, and even preserve the structure of the DC decomposition and problem formulation (12), but at the expense of an extra set of 0-1 variables, which model the constraints that $\mu(A) = \mu(A \setminus \{i\})$ for at least one $i \in A$ for all A with cardinality greater than k [57], and this becomes expensive numerically.

Here we follow the approach from [18] called k-interactivity. To avoid an excessive number of monotonicity constraints, the values of the capacity are fixed at subsets of cardinality larger than k in such a way as to maximize its partial entropy. This differs to k-additive and k-maxitive capacities, which fix the Möbius and possibilistic Möbius values respectively. The inputs' interactions in subsets of cardinality larger than k are no longer zero, but are determined from the subsets of smaller cardinality, and maximizing the entropy allows each input to be accounted for to the maximal extent.

Maximizing the Shannon entropy is suitable in the Choquet integral setting, resulting in equidistant values of μ (for subset cardinality greater than k), i.e., $\mu(B) = h_{n-k}$, for some $h_{n-k} \in [0, 1]$ for $|B| = k + 1$, and larger subset measures are calculated as $h_{n-k} + \frac{|A|-k-1}{n-k-1}(1 - h_{n-k})$, $|A| = k + 2, \ldots, n$. As an analogue of k-interactivity in the Sugeno integral setting here we maximize the Rényi min-entropy H_∞, which results in solving the problem

$$\min \max_{A, |A|>k} (\mu(A \cup \{i\}) - \mu(A)).$$

The optimal solution is $\mu(A) = 1$ for all $A, |A| > k$.

The resulting capacity is thus $k + 1$-maxitive. The Sugeno integral with respect to such a capacity is expressed as

$$S_\mu(\mathbf{x}) = \left(\bigvee_{A \subseteq N, |A| \leqslant k} (\min_{i \in A} x_i) \wedge \mu(A) \right) \bigvee x_{(n-k)}. \tag{13}$$

That is, the $n-k-1$ smallest components of $\mathbf{x}$ do not participate in the calculation, and no weighting is applied to the $n - k$-th smallest component.

The advantage of using this approach is a very significant reduction of the number of monotonicity constraints, and also of the number of variables. On the other hand, one can apply the DC decomposition (9), and solve an LP problem similar to (12) at Step 2 of the DC algorithm. For example, for a symmetric capacity, we only need to add the constraints $h_2 = h_3 = \ldots = h_{n-k} = 1$ in (12).

On the technical side, note that the same set of constraints is involved irrespective of the value $\mathbf{y}$ obtained at Step 1 of the DCA. As was the case with symmetric capacities, we have a sequence of similar LP problems at Step 2 that only differ in terms of the coefficients in the objective. Once again we can save computation time by making small updates to a generic LP model from iteration to iteration.

6. Computational experiments

6.1. Methods used

In this section we compare the performance of various numerical optimization methods when solving problem (6), i.e., assuming the symmetry of the capacity. Firstly, we consider the following generic derivative-free solvers ported/implemented in the [58] C library:

1. the Nelder–Mead (`NM`) simplex method [59] with bound constraints [60,61],
2. Powell's method `BOBYQA` – based on an iteratively constructed quadratic approximation of the objective [53],
3. `SBPLX`, a variant of Rowan's Subplex [62] method that uses Nelder–Mead on a sequence of subspaces.

All methods are derivative-free and support box constraints (which are needed in the $\boldsymbol{\delta}$ representation).

Next, we study a derivative-free method supporting arbitrary inequality constraints, which makes it particularly suitable for minimizing F with respect to un-transformed $\mathbf{h}$ variables, namely:

4. Powell's `LINCOA` which, like `BOBYQA`, uses a quadratic approximation of the objective [51].

Here we used our own C port of Powell's original Fortran code available at `http://mat.uc.pt/~zhang/software.html#lincoa`.

Further, we use a method that explicitly relies on a DC decomposition of a non-smooth objective $F = G - H$:

5. the truncated codifferential (`CODIF`) method [52].

The original method (`http://napsu.karmitsa.fi/tcm/`) was implemented in Fortran, and we used `f2c` (`http://www.netlib.org/f2c/`) to prepare its C port.

Except for `LINCOA`, which has built-in support for the $h_1 = 1 \geqslant h_2 \geqslant h_3 \geqslant \ldots \geqslant h_n$ constraint, we tested the behavior of the aforementioned algorithms both in the case of the original variables with an L_1 penalty term for violating monotonicity of $\mathbf{h}$, a scaled version of

$$|h_1 - 1| + \sum_{i=2}^{n} \mathbf{1}(h_i > 1)|h_i - 1| + \sum_{i=2}^{n} \mathbf{1}(h_i < 0)|0 - h_i| + \sum_{i=2}^{n} \mathbf{1}(h_i > h_{i-1})|h_i - h_{i-1}|$$

and the changed variables $\delta_i \in [0, 1]$ (note that `CODIF` does not support box constraints, therefore we also used an additive penalty term here).

Finally, we considered up to 100 iterations of the DC algorithm (yet, the convergence was often observed much faster), where in Step 2 the minimization problem $\min G(\cdot) - \langle \cdot, \mathbf{y} \rangle$ was solved by means of:

- primal LP formulation (`lp_solve`, see `http://lpsolve.sourceforge.net/5.0/index.htm`),
- dual LP formulation (`lp_solve`),
- `LINCOA` (based on $\mathbf{h}$ variables with the monotonicity constraint),
- `BOBYQA`, `NM`, `SBPLEX`, and `CODIF` (based on $\boldsymbol{\delta}$ representation).

For each method, we considered the best solution out of 25 restarts from random initial parameter values.

6.2. Benchmark data

Benchmark data were generated as follows. In each scenario, for a given problem size $n \in \{5, 10, 25\}$, number of data vectors $K \in \{10, 100\}$ (for $n = 5$, also with $K \in \{25, 250\}$), and noise level $\sigma \in \{0, 0.1\}$, $\mathbf{h}$ was generated in such a way that each $h_i, i = 2, \ldots, n$ was sampled from the Beta distribution $B(p, q)$, where p and q followed the uniform distribution U on $[0.25, 5]$. Then $\mathbf{h}$ was sorted decreasingly. Each of the K data vectors $\mathbf{x}^{(j)}$ were also sampled from $B(p_j, q_j)$, with $p_j, q_j \sim U[0.25, 5]$. Finally, $y^{(j)}$ was set to be equal to $1 \wedge (S_{\mathbf{h}}(\mathbf{x}^{(j)}) + \varepsilon_j) \vee 0$, where ε_j follows the normal distribution $N(0, \sigma)$. Lastly, a tiny bit of noise (of order of magnitude 10^{-10}) was added to $y^{(j)}$ so as to ensure all the reference outputs were unique.

For each parameter triple (n, K, σ), $M = 100$ randomly generated data instances $(\mathbf{h}, \mathbf{X}, \mathbf{y})$ were considered.

The experiments were performed in the R environment [63], which was used to generate benchmark data scenarios, call the underlying solvers (which we implemented in C and C++) through the R/C API or Rcpp, gather and analyse the results.

For NLopt-based methods, we used the following built-in convergence criteria: *xtol_rel* of 10^{-7}, *stopval* of 10^{-10}, and *maxeval* of 100,000. In the case of the other methods, we relied on the default parameters as established by their authors.

6.3. Scenario 1: Outputs with no noise

First we consider the case of $\sigma = 0$, where the $\mathbf{h}$ vector generating each $y^{(j)}$ is the known solution to the optimization problem (with $F(\mathbf{h}) = 0$). Table 1 gives the fractions of cases where the algorithm found the true minimum (we assumed it fulfils $F(\mathbf{h}) < 10^{-6}K(K-1)/2$ to take into account numerical inaccuracies).

We get the highest overall performance in the case of the Nelder–Mead-type algorithms utilizing the $\boldsymbol{\delta}$-representation (`NM`, `SBPLX`). `LINCOA` (recall that it is the only generic derivative-free solver explicitly supporting the monotonicity constraint) also gives high overall accuracy, despite the fact that for larger n and K its performance dropped significantly. The `CODIF` ($\boldsymbol{\delta}$) method, relying on the explicit DC decomposition of the objective is also worth considering.

Generally the accuracy of all methods degrades as n and K increase. In particular only the $\boldsymbol{\delta}$-based `NM`, `SBPLX`, `CODIF` were able to indicate the optimal solution for $n = 25$ and $K = 100$. Increasing the number of restarts from 25 to some greater value, should lead to increases in accuracy for the other methods.

Almost all the DCA-based methods are characterized by sub-par performance. Taking into account their much higher run-times, we discourage their use in this application.

Note that even a single instance of the experiment (running the algorithm on one $(\mathbf{X}, \mathbf{Y})$ pair) required too much time for the two LP formulations of DCA with $K \geqslant 100$, due to $O(K^2)$ number of variables and constraints in the LP. Therefore, we omitted these scenarios from the experiments. However, we can note that despite promising results for $K = 10$, similar to the other DCA-based methods, here we might also expect a significant performance drop when K increases (as observed in the case $n = 5, K = 25$).

Finally, observe that even by random guessing (pick the minimum of the objective F at 25 randomly chosen $\mathbf{h}$), in the case of small K it is still possible to find a minimum in over 50% of cases. This provides some perspective on the performance of derivative-free approaches towards DCA.

Table 1. Case $\sigma = 0$; fraction of instances for which an algorithm found the optimal solution, $F(\mathbf{h}) \simeq 0$

n	5				10		25		Mean
K	10	25	100	250	10	100	10	100	
BOBYQA	0.98	0.99	1.00	1.00	0.86	0.63	0.80	0.01	0.78
BOBYQA ($\boldsymbol{\delta}$)	1.00	1.00	0.99	0.99	0.99	0.56	0.97	0.10	0.82
BOBYQA DCA ($\boldsymbol{\delta}$)	0.92	0.52	0.04	0.04	0.81	0.00	0.81	0.00	0.39
NM	1.00	1.00	1.00	1.00	0.92	0.74	0.86	0.14	0.83
NM ($\boldsymbol{\delta}$)	1.00	1.00	1.00	1.00	0.98	0.95	0.99	0.46	**0.92**
NM DCA ($\boldsymbol{\delta}$)	0.91	0.48	0.04	0.07	0.77	0.00	0.78	0.00	0.38
SBPLX	0.99	0.99	0.99	0.98	0.93	0.86	0.81	0.11	0.83
SBPLX ($\boldsymbol{\delta}$)	1.00	1.00	1.00	1.00	0.99	0.97	1.00	0.48	**0.93**
SBPLX DCA ($\boldsymbol{\delta}$)	0.88	0.42	0.03	0.01	0.78	0.00	0.79	0.00	0.36
LINCOA	1.00	0.99	1.00	1.00	0.97	0.96	0.97	0.23	0.89
LINCOA DCA	0.92	0.46	0.00	0.00	0.82	0.00	0.78	0.00	0.37
CODIF	0.95	0.99	1.00	1.00	0.92	0.64	0.84	0.00	0.79
CODIF ($\boldsymbol{\delta}$)	0.97	0.97	0.92	0.91	0.97	0.84	0.97	0.42	0.87
LP DCA (primal)	0.96	0.62	—	—	0.94	—	0.93	—	—
LP DCA (dual)	0.94	0.51	—	—	0.86	—	0.91	—	—
random guessing	0.71	0.14	0.01	0.00	0.43	0.00	0.50	0.00	0.22

6.4. Scenario 2: Outputs with noise

Next, in the case where $\sigma = 0.1$, the optimal solution is unknown. Hence, we decided to treat the (locally) best solution as found by all the algorithms as a reference and then check if each method had converged to a solution within the relative error of $< 1\%$. Entries in Table 2 give the fraction of occasions where the method obtained either the best solution or within 1%.

Here, the clear winner is the Nelder–Mead method with the $\boldsymbol{\delta}$ representation. `SBPLEX`, `LINCOA`, `CODIF` and `BOBYQA` also exhibit good accuracy.

As with the $\sigma = 0$ experiments, DCA-based methods proved sub-par.

7. Conclusions

In this paper we have developed methods for identifying capacities or fuzzy measures in the context of ordinal regression with respect to data aggregation using the Sugeno integral. The learning problem has been identified as one that can be formulated using the DC decomposition and hence benefit from DCA methods. Further to this, several sub-formulations were developed to take advantage of the Sugeno integral's structure in order to reduce the number of constraints required and so that different solving strategies could be incorporated at different levels of the optimization.

The algorithms were then compared in terms of accuracy under different conditions. Representing the unknown vector $\mathbf{h}$ in terms of iterated differences, which reduces the number of constraints required, was found to improve the likelihood of reaching the optimal solution with almost all the optimization approaches. DCA, although it does find solutions in the case of small dimension, did not prove to be very effective at reaching optimal solutions. However the `CODIF` method, which also uses a DC

Table 2. Case $\sigma = 0.1$; fraction of the cases where a method converged to the best solution as found by all the methods

n	5				10		25		Mean
K	10	25	100	250	10	100	10	100	
BOBYQA	0.42	0.54	0.99	1.00	0.13	0.26	0.01	0.00	0.42
BOBYQA ($\boldsymbol{\delta}$)	0.68	0.81	1.00	1.00	0.40	0.97	0.14	0.58	0.70
BOBYQA DCA ($\boldsymbol{\delta}$)	0.36	0.18	0.59	0.90	0.25	0.12	0.06	0.00	0.31
NM	0.69	0.91	1.00	1.00	0.31	0.77	0.01	0.01	0.59
NM ($\boldsymbol{\delta}$)	0.89	0.98	1.00	1.00	0.82	1.00	0.77	0.99	**0.93**
NM DCA ($\boldsymbol{\delta}$)	0.36	0.10	0.55	0.87	0.14	0.16	0.02	0.01	0.28
SBPLX	0.68	0.62	1.00	0.99	0.24	0.79	0.00	0.00	0.54
SBPLX ($\boldsymbol{\delta}$)	0.96	0.98	1.00	1.00	0.63	0.95	0.43	0.51	0.81
SBPLX DCA ($\boldsymbol{\delta}$)	0.28	0.08	0.46	0.86	0.15	0.05	0.02	0.00	0.24
LINCOA	0.84	0.92	1.00	1.00	0.70	0.98	0.35	0.58	0.80
LINCOA DCA	0.39	0.09	0.34	0.46	0.15	0.01	0.02	0.00	0.18
CODIF	0.62	0.87	1.00	1.00	0.27	0.45	0.03	0.00	0.53
CODIF ($\boldsymbol{\delta}$)	0.67	0.88	1.00	1.00	0.47	1.00	0.54	0.42	0.75
LP DCA (primal)	0.55	0.17	—	—	0.34	—	0.16	—	—
LP DCA (dual)	0.46	0.09	—	—	0.24	—	0.05	—	—

decomposition of the objective, shows relatively good performance.

There are a number of variations to the learning problem that could arise depending on the data available and the optimization objective. Here we have focused on the L_1 measured conflicts to an observed order, however we could also consider squared differences, minimization of the Kemeny distance or maximization of Spearman correlation.

The tools we have developed here contribute to our body of work on fitting the Sugeno integral to data [39–42]. Such methods allow broad application of this function to data modelling and decision making.

Acknowledgements

Marek Gagolewski wishes to acknowledge the support by the Czech Science Foundation through the project No. 18-06915S.

References

[1] Grabisch M. Set functions, games and capacities in decision making. Berlin, New York: Springer; 2016.
[2] Beliakov G, Bustince H, Calvo T. A practical guide to averaging functions. New York: Springer; 2016.
[3] Beliakov G, James S, Wu JZ. Discrete fuzzy measures. Computational aspects. Berlin, Heidelberg: Springer; 2019.
[4] Choquet G. Theory of capacities. Ann Inst Fourier. 1953;5:131–295.
[5] Klement EP, Mesiar R, Pap E. A universal integral as common frame for Choquet and Sugeno integral. IEEE Transactions on Fuzzy Systems. 2010;18(1):178–187.

[6] Grabisch M, Kojadinovic I, Meyer P. A review of methods for capacity identification in Choquet integral based multi-attribute utility theory: Applications of the Kappalab R package. European Journal of Operational Research. 2008;186(2):766–785.

[7] Beliakov G, James S, Li G. Learning Choquet-integral-based metrics for semisupervised clustering. IEEE Trans on Fuzzy Systems. 2011;19:562–574.

[8] Yager RR. Using fuzzy measures to construct multi-criteria decision functions. In: Soft computing based optimization and decision models. Springer; 2018. p. 231–239.

[9] Beliakov G. A new type of fuzzy integrals for decision making based on bivariate symmetric means. Int J Intelligent Syst. 2018;33:1660–1671.

[10] Beliakov G, James S, Wilkin T, et al. Robustifying OWA operators for aggregating data with outliers. IEEE Trans on Fuzzy Systems. 2018;26:1823–1832.

[11] Wu JZ, Yu LP, Li G, et al. Using the monotone measure sum to enrich the measurement of the interaction of multiple decision criteria. Journal of Intelligent and Fuzzy Systems. 2016;30(5):2529–2539.

[12] Wu JZ, Beliakov G. Probabilistic bipartition interaction index of multiple decision criteria associated with the nonadditivity of fuzzy measures. International Journal of Intelligent Systems. 2019;34:247–270.

[13] Shapley LS. A value for n-person games. Contributions to the Theory of Games. 1953; 2(28):307–317.

[14] Banzhaf J. Weight voting doesn't work: A mathematical analysis. Rutgers Law Review. 1965;19:317–343.

[15] Wu JZ, Beliakov G. Nonadditivity index and capacity identification method in the context of multicriteria decision making. Information Sciences. 2018;467:398–406.

[16] Wu JZ, Beliakov G. Nonmodularity index for capacity identifying with multiple criteria preference information. Information Sciences. 2019;492:164–180.

[17] Grabisch M. k-Order additive discrete fuzzy measures and their representation. Fuzzy Sets and Systems. 1997;92:167–189.

[18] Beliakov G, Wu JZ. Learning fuzzy measures from data: simplifications and optimisation strategies. Information Sciences. 2019;494:100–113.

[19] Marichal JL, Roubens M. Determination of weights of interacting criteria from a reference set. European Journal of Operational Research. 2000;124(3):641–650.

[20] Murillo J, Guillaume S, Bulacio P. k-Maxitive fuzzy measures: A scalable approach to model interactions. Fuzzy Sets and Systems. 2017;.

[21] Islam MA, Anderson DT, Pinar AJ, et al. Data-driven compression and efficient learning of the Choquet integral. IEEE Transactions on Fuzzy Systems. 2018;26(4):1908–1922.

[22] Mendez-Vazquez A, Gader P, Keller JM, et al. Minimum classification error training for Choquet integrals with applications to landmine detection. IEEE Transactions on Fuzzy Systems. 2008;16(1):225–238.

[23] Islam MA, Anderson DT, Petry F, et al. An efficient evolutionary algorithm to optimize the Choquet integral. International Journal of Intelligent Systems. 2019;34:366–385.

[24] Wu JZ, Zhang Q, Du Q, et al. Compromise principle based methods of identifying capacities in the framework of multicriteria decision analysis. Fuzzy Sets and Systems. 2014; 246:91–106.

[25] Meyer P, Roubens M. Choice, ranking and sorting in fuzzy multiple criteria decision aid. In: Figueira J, Greco S, Ehrogott M, editors. Multiple criteria decision analysis: State of the art surveys. Springer New York; 2005. p. 471–503.

[26] Angilella S, Bottero M, Corrente S, et al. Non additive robust ordinal regression for urban and territorial planning: An application for siting an urban waste landfill. Annals of Operations Research. 2016;245(1-2):427–456.

[27] Beliakov G. Construction of aggregation functions from data using linear programming. Fuzzy Sets and Systems. 2009;160:65–75.

[28] Tehrani A, Cheng W, Dembczynski K, et al. Learning monotone nonlinear models using the Choquet integral. Machine Learning. 2012;89(1-2):183–211.

[29] Anderson MF, Anderson DT, Wescott DJ. Estimation of adult skeletal age-at-death using

the Sugeno fuzzy integral. American Journal of Physical Anthropology. 2010;142(1):30–41.
[30] Anderson D, Keller J, Havens T. Learning fuzzy-valued fuzzy measures for the fuzzy-valued Sugeno fuzzy integral. Lecture Notes in Artificial Intelligence. 2010;6178:502–511.
[31] Brabant Q, Couceiro M. k-Maxitive Sugeno integrals as aggregation models for ordinal preferences. Fuzzy Sets and Systems. 2018;343:65 – 75.
[32] Dubois D, Marichal JL, Prade H, et al. The use of the discrete Sugeno integral in decision-making: A survey. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 2001;9(05):539–561.
[33] Marichal JL. An axiomatic approach of the discrete Sugeno integral as a tool to aggregate interacting criteria in a qualitative framework. IEEE Transactions on Fuzzy Systems. 2001;9(1):164–172.
[34] Zadeh L. The concept of a linguistic variable and its application to approximate reasoning. Part I. Inf Sciences. 1975;8:199–249.
[35] Zadeh L. The concept of a linguistic variable and its application to approximate reasoning. Part II. Inf Sciences. 1975;8:301–357.
[36] Zadeh L. The concept of a linguistic variable and its application to approximate reasoning. Part III. Inf Sciences. 1975;9:43–80.
[37] Delgado M, Verdegay JL, Vila M. On aggregation operations of linguistic labels. Int J Intelligent Systems. 1993;8:351–370.
[38] Keller JM, Osborn J. Training the fuzzy integral. International Journal of Approximate Reasoning. 1996;15(1):1 – 24.
[39] Gagolewski M, James S. Fitting symmetric fuzzy measures for discrete Sugeno integration. In: Advances in intelligent systems and computing. Vol. 642. Springer; 2018. p. 104–116.
[40] Gagolewski M, James S, Beliakov G. Supervised learning to aggregate data with the Sugeno integral. IEEE Transactions on Fuzzy Systems. 2019;27(4):810–815.
[41] Beliakov G, Gagolewski M, James S. Robust fitting for the Sugeno integral with respect to general fuzzy measures. Information Sciences. 2019;Under Review.
[42] Beliakov G, Gagolewski M, James S. Aggregation on ordinal scales with Sugeno integral for biomedical applications. Information Sciences. 2019;501:377–387.
[43] Grabisch M. The representation of importance and interaction of features by fuzzy measures. Pattern Recognition Letters. 1996;17(6):567–575.
[44] Angilella S, Greco S, Matarazzo B. Non-additive robust ordinal regression: A multiple criteria decision model based on the Choquet integral. European Journal of Operational Research. 2010;201(1):277–288.
[45] Corrente S, Greco S, Kadziński M, et al. Robust ordinal regression in preference learning and ranking. Machine Learning. 2013;93(2-3):381–422.
[46] Corrente S, Greco S, Ishizaka A. Combining analytical hierarchy process and Choquet integral within non-additive robust ordinal regression. Omega. 2016;61:2–18.
[47] Bagirov A, Karmitsa N, Makela M. Introduction to non-smooth optimization: Theory, practice and software. Springer, Berlin, Heidelberg; 2014.
[48] Horst R, Thoai N. DC programming: Overview. J Optim Theory and Appl. 1999;103:1 – 43.
[49] Ferrer A, Bagirov A, Beliakov G. Solving DC programs using the cutting angle method. Journal of Global Optimization. 2015;61:71–89.
[50] Le Thi Hoai An, Le Hoai Minh, Pham Dinh Tao. New and efficient DCA based algorithms for minimum sum-of-squares clustering. Pattern Recognitions. 2004;47:388–401.
[51] Powell M. On fast trust region methods for quadratic models with linear constraints. Mathematical Programming Computation. 2015;7:237–267.
[52] Bagirov A, Ugon J. Codifferential method for minimizing nonsmooth DC functions. Journal of Global Optimization. 2011;50:3–22.
[53] Powell M. The BOBYQA algorithm for bound constrained optimization without derivatives. DAMTP NA2009/06, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge CB3 9EW, UK; 2009.
[54] Calvo T, De Baets B. Aggregation operators defined by k-order additive/maxitive fuzzy

measures. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 1998;6(6):533–550.

[55] Mesiar R. k-Order additive fuzzy measures. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 1999;7(6):561–568.

[56] Mesiar R. Generalizations of k-order additive discrete fuzzy measures. Fuzzy sets and systems. 1999;102(3):423–428.

[57] Beliakov G, Wu JZ. Learning k-maxitive fuzzy measures from data by mixed integer programming. Under Review. 2018;.

[58] Johnson SG. The NLopt nonlinear-optimization package ; 2019. Http://ab-initio.mit.edu/nlopt.

[59] Nelder JA, Mead R. A simplex algorithm for function minimization. Computer Journal. 1965;7(4):308–313.

[60] Box MJ. A new method of constrained optimization and a comparison with other methods. Computer Journal. 1965;8(1):42–52.

[61] Richardson JA, Kuester JL. The complex method for constrained optimization. Communications of the ACM. 1973;16(8):487–489.

[62] Rowan T. Functional stability analysis of numerical algorithms [dissertation]. Department of Computer Sciences, University of Texas at Austin; 1990.

[63] R Development Core Team. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; 2019. Http://www.R-project.org/.