Macsum aggregation learning

Yassine Hmidy, Agnès Rico, Olivier Strauss

Macsum aggregation learning

Yassine Hmidy[a], Agnès Rico[b], Olivier Strauss[a]

[a] *LIRMM, Université de Montpellier, CNRS, France*
[b] *LIRIS, Université Claude Bernard Lyon 1, CNRS, France*

Abstract

Identification of a multi-input/single-output system enable the creation of a mathematical model that as accurately as possible represents the input-output relation induced by a physical system. When there is little information about the physical laws related to the system or when the system is too complex, methods such as parametric identification are used to define the system model. In that case, preliminary assumptions can be made about the system leading to a parametric aggregation function based model. This parametric model may be learned by estimating the parameters of this aggregation function from a representative set of inputs/outputs. A major difficulty is to design a model that is relatively simple yet precise enough to meet the user's needs. Linear models are commonly used because they meet these two contradictory constraints. However, use of a linear model is often at the expense of the accuracy of input-output relationship description.

In a recent paper, a new modeling approach was proposed, under the name of macsum modeling [31], which aims at replacing the linear model concept by a set of linear models. The obtained aggregation function leads to an interval-valued output. This represents the lack of accuracy of the model to predict the system output when the inputs are know. An interesting feature of this model is that it is ruled by a single precise parametric vector whereas the output is imprecise. Moreover, the vector dimension is equal to that of the input space.

In this paper, we propose a method to learn such a model so that it best reflects the input-output relationship of the considered system. This method is based, as in the case of linear modeling, on a regression method. This is particularly new approach because the macsum aggregation is based on a Choquet integral and very few authors have proposed learning of an aggregation function based on the Choquet integral.

Keywords: Imprecise linear system, Choquet integral, non-additive aggregation, non-monotonic set functions, regression, learning

Email addresses: `Yassine.Hmidy@lirmm.fr` (Yassine Hmidy), `Agnes.Rico@univ-lyon1.fr` (Agnès Rico), `Olivier.Strauss@lirmm.fr` (Olivier Strauss)

1. Introduction

Aggregation functions are quite commonly used to model the behavior of a multi-input/single output (MISO) physical system. A physical system can be seen as a process whereby known or measured inputs are transformed into a potentially measurable output (e.g. weather prediction, medical diagnosis, automatic driving systems, etc.). Use of an aggregation function is quite relevant since it groups several input values to form a single output value. The most common learning method involves determining an aggregation function as a model that mimics the behavior of the considered system [14]. Most aggregation functions used in this context are parametric, i.e. their behavior is regulated by a set of values called "function parameters" or **kernel** of the function.

In many cases, the inherent lack of information on a system, or measuring its inputs or output, makes it difficult to perfectly comprehend the input-output relationship.

Several approaches have been proposed in the relevant literature to account for this weakness, with the aim of obtaining aggregation functions that yield an interval-valued rather than a precise-valued output [11, 22, 1, 23, 3]. Among all these approaches, some authors have proposed to represent the input-output relationship of a system by a convex set of aggregation functions [35, 6, 21, 16, 18, 23, 32, 28, 17, 2]. The approach we propose in [31] is in this category. It consists of representing a convex family of linear functions by an interval-valued aggregation based on a set function, called the *macsum* operator, which is ruled by a single precise valued kernel. In the above-mentioned article, we have assessed a large of properties of the macsum set function and their consequences on the macsum (interval-valued) aggregation function. The interval-valued output of this aggregation function turns out to be the convex set of all single-valued outputs of the represented linear aggregation functions. It thus can be viewed as an interval-valued linear aggregation.

The remaining question to address is "what value to give to the kernel for this aggregation function to best represent the system behavior?". We propose to partly answer this question in this article. We show how it is possible to use, as in any function learning, a regression method to estimate the kernel allowing the aggregation function to best represent the considered system behavior. One of the main difficulties of the proposed approach concerns the fact that the aggregation function we propose is based on a Choquet integral.

Very few articles in the abundant literature on learning have proposed to learn a model based on a Choquet integral. Among these few articles, one of the most relevant to our proposed approach is that of Tehrani et al. [13] that describes how to learn monotone nonlinear models based on a Choquet integral for classification purposes. These authors modelled input-output relationship using a capacity, scaling parameter and threshold. The learning process consists of maximizing a log-likelihood to find optimal values of the Mœbius transform of the capacity and the two parameters. High complexity is an issue regarding this approach, (i.e. high number of parameters – 2^N parameters, with N being the number of inputs – of the model). This issue is addressed by focussing solely

on the k-additive capacities and finding a suitable value for k [12]. In contrast to this type of approach, Havens et al. [15] propose to learn a MISO model based on a Choquet integral and a non-monotonic set function. They propose to learn 2^N weights associated with the capacity (with N being the number of inputs) by proposing a matrix formulation of the problem and by performing a least squares minimization, i.e. a method they call Choquet integral regression.

The approach we describe here can be compared to that of [15] since we propose to learn the parameters of a non-monotonic set-function. However, this set-function is ruled by a number of parameters equal to the number of inputs. It also differs in that we do not consider using the Choquet integral as a tool for learning monotonic nonlinear models but rather for learning a convex set of linear models, i.e. a model that aims at representing an imprecise input-output relationship. This difference makes the resulting model more easily interpretable.

After this introduction, we propose, in Section 2, a set of notations and definitions to facilitate reading of the article. We also position our work in relation to the literature. In Section 3, we summarize a certain number of important properties of the macsum operator that are necessary to understand its use in system modeling. In Section 4, we introduce the operator-based aggregation notion which is a generalization of linear aggregation notion. We present a new formulation of the macsum aggregation function defined in [31]. Section 5 proposes a regression technique to adjust the parameter vector of the macsum aggregation function based on a set of input-output pairs of the system to model. In Section 6, we illustrate the properties of the obtained aggregation function with an image processing experiment. We finally conclude in Section 7.

2. Preliminary aspects

2.1. Notations

- $\Omega = \{1, \ldots, N\} \subset \mathbb{N}$.
- A vector of Ω is a function $\boldsymbol{x} : \Omega \to \mathbb{R}$ defined by a discrete subset of $\mathbb{R}^{\mathbb{N}}$ denoted $\boldsymbol{x} = (x_1, \cdots, x_N)$.
- A parametric function can be denoted $f_{\boldsymbol{\varphi}}(.)$ or $f(., \varphi)$, with $\boldsymbol{\varphi}$ being a vector of Ω called a **kernel**.
- The set of kernels of Ω is denoted $\mathcal{K}(\Omega)$.
- A **maxitive** kernel of Ω is a kernel $\boldsymbol{\pi} \in \mathcal{K}(\Omega)$ such that $\forall i \in \Omega$, $\pi_i \in [0, 1]$ and $\max_{i \in \Omega} \pi_i = 1$.
- $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ being a kernel, we define two kernels, $\boldsymbol{\varphi}^+$ and $\boldsymbol{\varphi}^-$, by $\forall i \in \Omega$ $\varphi_i^+ = \max(0, \varphi_i)$ and $\varphi_i^- = \min(0, \varphi_i)$.
- $\lceil . \rceil$ is a permutation that sorts $\boldsymbol{\varphi}$ in increasing order: $\varphi_{\lceil 1 \rceil} \leq \varphi_{\lceil 2 \rceil} \leq \cdots \leq \varphi_{\lceil N \rceil}$.

- $\lfloor . \rfloor$ is a permutation that sorts $\boldsymbol{\varphi}$ in decreasing order: $\varphi_{\lfloor 1 \rfloor} \geq \varphi_{\lfloor 2 \rfloor} \geq \cdots \geq \varphi_{\lfloor N \rfloor}$.
- $\mathbb{IR}$ is the set of real intervals.

  $[y] = [\underline{y}, \overline{y}]$ is a real interval whose lower bound is $\underline{y}$ and upper bound is $\overline{y}$.

2.2. Definitions

- Let $\mathcal{X} = \{\boldsymbol{x}^k\}_{k \in \{1,..,M\}}$ be a set of vectors of $\mathbb{R}^N$ and $\mathcal{Y} = \{y_k\}_{k \in \{1,..,M\}}$ be a set of real numbers. We define a dataset as being a set of pairs $\mathcal{X} \times \mathcal{Y} = \left\{\boldsymbol{x}^k, y_k\right\}_{k \in \{1,..,M\}}$.
- A binary relation $\mathcal{R}$ over sets $\mathcal{X}$ and $\mathcal{Y}$ is a new set of ordered pairs $(\boldsymbol{x}, y)$ with $\boldsymbol{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $\boldsymbol{x}\mathcal{R}y$, which means that $\boldsymbol{x}$ is in relation with y.
- To a binary relation $\mathcal{R}$ is associated a graph $\mathcal{G}_{\mathcal{R}}$ which is a subset of $\mathbb{R}^N \times \mathbb{R}$ containing all the pairs of $\mathcal{R}$-related elements. That is to say $(\boldsymbol{x}, y) \in \mathcal{G}_{\mathcal{R}}$ if and only if $\boldsymbol{x}\mathcal{R}y$.
- A set function is a function $\mu : 2^{\Omega} \to \mathbb{R}$ that maps any subset of Ω onto a real value. To a set function μ is associated its complementary set function μ^c: $\forall A \subseteq \Omega$, $\mu^c(A) = \mu(\Omega) - \mu(A^c)$. Moreover, $\mu(\varnothing) = \mu^c(\varnothing) = 0$.
- A set function μ is concave or **supermodular** if , $\forall A, B \subseteq \Omega$, $\mu(A \cup B) + \mu(A \cap B) \leq \mu(A) + \mu(B)$.
- A set function μ is convex or **submodular** if , $\forall A, B \subseteq \Omega$, $\mu(A \cup B) + \mu(A \cap B) \geq \mu(A) + \mu(B)$.
- A set function μ is additive if , $\forall A, B \subseteq \Omega$, $\mu(A \cup B) + \mu(A \cap B) = \mu(A) + \mu(B)$. By definition, an additive set function is both concave and convex.
- A set function μ is maxitive if , $\forall A, B \subseteq \Omega$, $\mu(A \cup B) = \max(\mu(A), \mu(B))$.
- The asymmetric discrete Choquet integral w.r.t. a set function μ [25], denoted $\check{\mathbb{C}}_{\mu}$, is defined by:

$$\check{\mathbb{C}}_{\mu}(\boldsymbol{x}) = \sum_{k=1}^{N} x_{(k)}.(\mu(A_{(k)}) - \mu(A_{(k+1)})) = \sum_{k=1}^{N} (x_{(k)} - x_{(k-1)}).\mu(A_{(k)}),$$

  with (.) being the permutation that sorts elements of $\boldsymbol{x}$ in increasing order: $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$ with $x_{(0)} = 0$ and $A_{(i)}$ $(i \in \Omega)$ being the ith coalition of Ω defined by $A_{(i)} = \{(i), \ldots, (N)\}$ with $A_{(N+1)} = \varnothing$.

- A capacity is a set function $\upsilon : 2^{\Omega} \to \mathbb{R}^+$, monotonic w.r.t. set inclusion: $\forall A \subseteq B \subseteq \Omega$, $\upsilon(A) \leq \upsilon(B)$ and such that $\upsilon(\Omega) = 1$.

2.3. Related work

A MISO[1] real system $\mathcal{S}$ is a process that generates an output $y \in \mathbb{R}$ from any given input vector $\boldsymbol{x} \in \mathbb{R}^N$. It can be seen as a binary relation $\mathcal{R}$ of $\mathbb{R}^N \times \mathbb{R}$ whose graph is denoted $\mathcal{G}_{\mathcal{R}}$. The goal of learning a model of this system is to determine its output for any input vector as accurately as possible. In other words, the goal is to know every pair of $\mathcal{G}_{\mathcal{R}}$ given we already know a subset of M pairs of $\mathcal{G}_{\mathcal{R}}$ denoted $\mathcal{X} \times \mathcal{Y}$.

The most commonly used method involves proposing a parametric model in the form of a parametric aggregation function $f_{\boldsymbol{\varphi}}$ that outputs a real value $y = f_{\boldsymbol{\varphi}}(\boldsymbol{x})$ for any element $\boldsymbol{x} \in \mathbb{R}^N$. Learning this function, based on the $\mathcal{X} \times \mathcal{Y}$ dataset, simply involves finding a kernel $\hat{\boldsymbol{\varphi}} \in \mathcal{K}(\Omega)$ such that, given any element $(\boldsymbol{x}, y) \in \mathcal{G}_{\mathcal{R}}$, this function would provide an estimate $\hat{y} = f_{\hat{\boldsymbol{\varphi}}}(\boldsymbol{x})$ that is as close as possible to y. This so-called regression process usually consists of minimizing a distance d w.r.t. $\boldsymbol{\varphi}$. This can be written as [14]:

$$\hat{\boldsymbol{\varphi}} = \underset{\boldsymbol{\varphi}}{\operatorname{argmin}} \left(\sum_{k=1}^{M} d\left(f_{\boldsymbol{\varphi}}(\boldsymbol{x}^k), y_k \right) \right).$$

However, in many cases, the use of a single function is insufficient to model the overall complexity of a real system. This is why several scientists have tried to introduce the notion of imprecision in the modeling of MISO systems. What these new aggregation functions represent and how their authors propose to learn these functions is the subject of this section. We subdivide the approaches proposed in the relevant literature into three categories, depending on whether they intend to focus on a defect in the acuity of the prediction, in the structural accuracy of the data, or in the consistency of the system.

2.3.1. Prediction uncertainty

This first category includes methods to assess the uncertainty concerning the ability of a model to accurately forecast the output of a system. This can be ascribed to random variations in the system output measurements. Such an approach is a major focus of the scientific community since the emergence of inferential statistics and its implication in learning processes [14]. The difficulty of accurately representing a system can be expressed by replacing the precise output value yield of the model by an interval-valued output. Most methods outlined in the literature aim at creating a confidence interval of the model output values. The article of Efron [11] provides a good survey of most existing methods. In a less classical way, Jaulin and al. propose methods that involve creating guaranteed intervals using interval analysis [22].

2.3.2. Data imprecision

The second category groups methods that choose, as a model, an aggregation function that produces intervals. This kind of modeling seems appropriate

[1] multi input - single output

since it is highly relevant to use imprecise data in many areas where there is data measurement imprecision (inputs or outputs) for a system. An approach widely used to learn the parameters of such a model involves designing a linear regression that tends to minimize the distance between the midpoint of the desired output and that of the interval-valued output of the model [1, 23]. Other approaches have been described based on adding, to the previous approach, a second regression that aims to minimize the distance between the spread of the desired interval-valued outputs and that of the interval valued output of the model [3].

2.3.3. System incoherence

The third category includes methods that attempt to design a model that accounts for the incoherence of a system. A system is said to be incoherent when it generates different outputs for the same input. Such a situation can occur when a system is not translational invariant (spatially, e.g. camera retina, temporally, e.g. disintegration process, etc.) or because some unknown (unmeasured) inputs modify the system behavior. In this case, the imprecision is intrinsic to the system. It thus makes sense that the model should reflect this imprecision. The system incoherence can be addressed by replacing the modeling using a single aggregation function by a modeling using a set of aggregation functions. Moreover, this set of functions would intuitively seem to be convex, i.e. if two aggregation functions f_1 and f_2 belong to the considered set, then $\forall \alpha \in [0,1]\ \alpha.f_1 + (1-\alpha).f_2$ also belongs to the set. The output of such a model would intuitively seem to be convex i.e. if $\boldsymbol{x}\mathcal{R}y_1$ and $\boldsymbol{x}\mathcal{R}y_2$ then $\forall \alpha \in [0,1]\ \boldsymbol{x}\mathcal{R}(\alpha.y_1 + (1-\alpha)y_2)$.

The main approach proposed in the literature, to address the incoherence, consists of considering a parametric function whose parameter is imprecise. Learning this kind of models involves regression analysis [35], which is a data-based functional relationship [6, 21, 16, 18, 23]. The learning process for interval-valued parameters has also been formulated as a linear [19] or quadratic [33] programming problem.

An extension of this crisp approach has also been proposed with the kernel being composed of fuzzy numbers [32, 28, 17] or fuzzy intervals [2]. These fuzzy parameters are learned by minimizing a cost that is a fuzzy version of the quadratic cost [5]. Recently, artificial neural networks have been modified to perform fuzzy regression, with the back-propagation algorithm being adapted to the case where the weights are fuzzy numbers [30, 26].

2.4. Imprecise linear approach

Our proposed approach is in this third category, with the aim of representing incoherence within the system by focusing on linear systems. A system is said to be linear if, for any pair $((\boldsymbol{x}, y), (\boldsymbol{x}', y')) \in \mathcal{G}_{\mathcal{R}} \times \mathcal{G}_{\mathcal{R}}$, we have $(\boldsymbol{x}+\boldsymbol{x}', y+y') \in \mathcal{G}_{\mathcal{R}}$ i.e $(\boldsymbol{x}+\boldsymbol{x}')\mathcal{R}(y+y')$. In that case, the aggregation function $f_{\boldsymbol{\varphi}}$ reduces to a sum of elements of the input vector $\boldsymbol{x}$ weighted by elements of the kernel $\boldsymbol{\varphi}$: $y = f_{\boldsymbol{\varphi}}(\boldsymbol{x}) = \sum_{i=1}^{N} x_i.\varphi_i$.

Imprecise linear approches provide a way to deal with incoherent systems while keeping the simplicity of a linear model. An imprecise linear model of a MISO system has an imprecise kernel. It can be formulated as:

$$[y] = f_{[\boldsymbol{\varphi}]}(\boldsymbol{x}) = \{f_{\boldsymbol{\psi}}(\boldsymbol{x}) \mid \boldsymbol{\psi} \in [\boldsymbol{\varphi}]\}, \tag{1}$$

with $[y] \in \mathbb{IR}$, $[\boldsymbol{\varphi}] \in \mathbb{IR}^N$ and $\boldsymbol{x} \in \mathbb{R}^N$. $[y]$ is the interval of all outputs that would have been obtained by considering $f_{\boldsymbol{\psi}}(\boldsymbol{x})$ with $\boldsymbol{\psi} \in [\boldsymbol{\varphi}]$. The linearity of the aggregation function f induces that, with $[\boldsymbol{\varphi}]$ being convex, the output of $f_{[\boldsymbol{\varphi}]}(\boldsymbol{x})$ is also convex, i.e. is an interval of $\mathbb{R}$.

One of the weaknesses of this type of model is that it is impossible to represent a set of linear functions with the same gain[2]. This gain is equal to the sum of the kernel elements.

In [31], we propose such an imprecise linear model. The macsum aggregation function we define outputs the set of real values that should have been obtained by considering a convex set of linear functions having the same gain. This set is completely defined by a single kernel $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$. In the following section, we summarize the basic principles of this aggregation function whose input vector and kernel are precise but whose output is interval valued.

3. Aggregation operators

This section includes many proposals from previous studies [31].

Let $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ be a kernel.We define an operator as being a kernel-based set function, i.e. a set function $\mu_{\boldsymbol{\varphi}} : 2^{\Omega} \to \mathbb{R}$ that maps any subset $A \subseteq \Omega$ onto a real value, with this value being obtained by systematic computation involving the values of the kernel $\boldsymbol{\varphi}$.

As in the linear case, $\mu_{\boldsymbol{\varphi}}(\Omega) = \mu^c_{\boldsymbol{\varphi}}(\Omega)$ value could be considered as the **gain of the aggregation operator** $\mu_{\boldsymbol{\varphi}}$, as we call it hereafter.

The most common operator is the **linear operator** $\lambda_{\boldsymbol{\varphi}}$ defined by:

$$\forall A \subseteq \Omega, \lambda_{\boldsymbol{\varphi}}(A) = \sum_{i \in A} \varphi_i.$$

Being additive, its complementary operator equals the operator:

$$\forall A \subseteq \Omega, \lambda^c_{\boldsymbol{\varphi}}(A) = \sum_{i \in \Omega} \varphi_i - \sum_{i \in A^c} \varphi_i = \sum_{i \in A} \varphi_i.$$

The **macsum operator** $\nu_{\boldsymbol{\varphi}}$ and its complementary operator $\nu^c_{\boldsymbol{\varphi}}$, introduced in [31], are defined as $\forall A \subseteq \Omega$:

$$\nu_{\boldsymbol{\varphi}}(A) = \max_{i \in A} \varphi_i^+ + \min_{i \in \Omega} \varphi_i^- - \min_{i \in A^c} \varphi_i^-, \tag{2}$$

$$\nu^c_{\boldsymbol{\varphi}}(A) = \min_{i \in A} \varphi_i^- + \max_{i \in \Omega} \varphi_i^+ - \max_{i \in A^c} \varphi_i^+. \tag{3}$$

[2]The gain of a linear function is the multiplicative coefficient by which a constant input would be multiplied.

As shown in [31], the macsum operator is a concave set function and its complementary is a convex set function.

Hereafter, it should be noted that if $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ is a maxitive kernel, then $\nu_{\boldsymbol{\varphi}}$ is equivalent to a possibilty measure [8], i.e. a maxitive capacity, usually denoted $\Pi_{\boldsymbol{\varphi}}$, and $\nu^c_{\boldsymbol{\varphi}}$ is equivalent to a necessity measure, usually denoted $N_{\boldsymbol{\varphi}}$. This stems from the fact that, where $\boldsymbol{\varphi}$ is a maxitive kernel, $\boldsymbol{\varphi}^+ = \boldsymbol{\varphi}$ and $\boldsymbol{\varphi}^- = \mathbf{0}$, with $\mathbf{0}$ being the vector whose elements are all zero. Moreover $\Pi_{\boldsymbol{\varphi}}(\Omega) = N_{\boldsymbol{\varphi}}(\Omega) = 1$.

Let $\boldsymbol{\varphi}, \boldsymbol{\psi} \in \mathcal{K}(\Omega)$ be two kernels of Ω, then we say that the kernel $\boldsymbol{\varphi}$ **dominates** the kernel $\boldsymbol{\psi}$ iff $\forall A \subseteq \Omega$, $\nu^c_{\boldsymbol{\varphi}}(A) \leq \lambda_{\boldsymbol{\psi}}(A) \leq \nu_{\boldsymbol{\varphi}}(A)$ (i.e. the set function $\nu_{\boldsymbol{\varphi}}$ dominates the set function $\lambda_{\boldsymbol{\psi}}$).

We define the core of a kernel $\boldsymbol{\varphi}$ as the subset $\mathcal{M}(\boldsymbol{\varphi}) \in \mathcal{K}(\Omega)$ of the kernels of Ω that are dominated by $\boldsymbol{\varphi}$:

$$\mathcal{M}(\boldsymbol{\varphi}) = \{\boldsymbol{\psi} \in \mathcal{K}(\Omega) \ / \ \forall A \subseteq \Omega, \ \nu^c_{\boldsymbol{\varphi}}(A) \leq \lambda_{\boldsymbol{\psi}}(A) \leq \nu_{\boldsymbol{\varphi}}(A)\}.$$

The two following properties – proven in [31] – are mandatory w.r.t. the learning application we propose:

- $\forall \boldsymbol{\varphi} \in \mathcal{K}(\Omega)$, $\exists \boldsymbol{\psi} \in \mathcal{K}(\Omega)$ such that $\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi})$, i.e. $\mathcal{M}(\boldsymbol{\varphi}) \neq \varnothing$.
- $\forall \boldsymbol{\psi} \in \mathcal{K}(\Omega)$ $\exists \boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ such that $\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi})$.

4. Operator-based aggregation

This section reviews some important results of [31]. Operator-based aggregations are set functions that output interval-valued results. The macsum aggregation and the linear aggregation are two examples of such aggregations.

4.1. Interval-valued aggregation

Let $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ be a kernel of Ω, $\mu_{\boldsymbol{\varphi}}$ a concave operator and $\boldsymbol{x} \in \mathbb{R}^N$ a real vector, then we define $\mathcal{A}_\mu : \mathbb{R}^N \times \mathcal{K}(\Omega) \to \mathbb{IR}$ as being a μ-interval-valued aggregation function. It associates, to any vector $\boldsymbol{x} \in \mathbb{R}^N$, a real interval $[y] \in \mathbb{IR}$ via the weighting sequence defined by the kernel $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ by: $[y] = [\underline{y}, \overline{y}] = \mathcal{A}_\mu(\boldsymbol{x}, \boldsymbol{\varphi})$ with $\underline{y} = \underline{\mathcal{A}}_\mu(\boldsymbol{x}, \boldsymbol{\varphi}) = \check{\mathbb{C}}_{\mu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x})$ and $\overline{y} = \overline{\mathcal{A}}_\mu(\boldsymbol{x}, \boldsymbol{\varphi}) = \check{\mathbb{C}}_{\mu_{\boldsymbol{\varphi}}}(\boldsymbol{x})$, where $\check{\mathbb{C}}$ stands for the asymmetric Choquet integral.

4.2. Linear aggregation

Let $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ be a kernel of Ω, then the linear aggregation is defined by using the linear operator $\lambda_{\boldsymbol{\varphi}}$. Since $\lambda_{\boldsymbol{\varphi}} = \lambda^c_{\boldsymbol{\varphi}}$, $\check{\mathbb{C}}_{\lambda^c_{\boldsymbol{\varphi}}} = \check{\mathbb{C}}_{\lambda_{\boldsymbol{\varphi}}}$ thus $\overline{\mathcal{A}}_\lambda(\boldsymbol{x}, \boldsymbol{\varphi}) = \underline{\mathcal{A}}_\lambda(\boldsymbol{x}, \boldsymbol{\varphi}) = y$ and therefore $\mathcal{A}_\lambda(\boldsymbol{x}, \boldsymbol{\varphi}) = [y, y]$ is a degenerate interval, i.e. a real value.

Proposition 4.1. *The linear aggregation amounts to calculating the sum of the vector $\boldsymbol{x}$ values weighted by the vector $\boldsymbol{\varphi}$ values: $\mathcal{A}_\lambda(\boldsymbol{x}, \boldsymbol{\varphi}) = \sum_{i=1}^N x_i.\varphi_i$.*

The proof of this property is trivial. It is based on the fact that the Choquet integral reduces to a simple integral when the used measure is linear.

4.3. Macsum aggregation

Let $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ be a kernel of Ω, the macsum aggregation is defined by using the macsum operator $\nu_{\boldsymbol{\varphi}}$: $\mathcal{A}_\nu(\boldsymbol{x}, \boldsymbol{\varphi}) = [y] = [\underline{y}, \overline{y}] = [\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x}), \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x})]$.

The values of $\underline{y}$ and $\overline{y}$ are given by [31]:

$$\overline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(x) = \sum_{k=1}^{N}(x_{(k)} - x_{(k-1)}).\left(\max_{i=k}^{N}\varphi^+_{(i)} + \min_{i=1}^{N}\varphi^-_i - \min_{i=1}^{k-1}\varphi^-_{(i)}\right), \quad (4)$$

$$\underline{y} = \check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(x) = \sum_{k=1}^{N}(x_{(k)} - x_{(k-1)}).\left(\min_{i=k}^{N}\varphi^-_{(i)} + \max_{i=1}^{N}\varphi^+_i - \max_{i=1}^{k-1}\varphi^+_{(i)}\right), \quad (5)$$

where (.) is a permutation that sorts vector $\boldsymbol{x}$ in increasing order $(x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)})$, $x_{(0)} = 0$.

The following lemma was proved by Rico and Dubois in [10] by using the Mœbius transform of a possibility measure associated with a discrete possibility distribution to give another form to the discrete Choquet integral w.r.t. this possibility measure.

Lemma 4.2. *Let $\boldsymbol{\pi}$ be a maxitive kernel of $\mathcal{K}(\Omega)$. Let $\Pi_{\boldsymbol{\pi}}$ be the maxitive capacity (possibility measure) defined by $\forall A \in \Omega$, $\Pi_{\boldsymbol{\pi}}(A) = \max_{i\in A}\pi_i$, then*

$$\begin{aligned}
\check{\mathbb{C}}_{\Pi_{\boldsymbol{\pi}}}(\boldsymbol{x}) &= \sum_{k=1}^{N}(x_{(k)} - x_{(k-1)}).\Pi_{\boldsymbol{\pi}}(A_{(k)}) = \sum_{k=1}^{N}(x_{(k)} - x_{(k-1)}).\max_{i=k}^{N}\pi_{(i)} \\
&= \sum_{k=1}^{N}(\pi_{\lfloor k\rfloor} - \pi_{\lfloor k+1\rfloor})\max_{i=1}^{k} x_{\lfloor i\rfloor}, \text{ and} \\
\check{\mathbb{C}}_{\Pi^c_{\boldsymbol{\pi}}}(\boldsymbol{x}) &= \sum_{k=1}^{N}(x_{(k)} - x_{(k-1)}).\Pi^c_{\boldsymbol{\pi}}(A_{(k)}) = x_{(N)} - \sum_{k=1}^{N}(x_{(k)} - x_{(k-1)}).\max_{i=1}^{k-1}\pi_{(i)} \\
&= \sum_{k=1}^{N}(\pi_{\lfloor k\rfloor} - \pi_{\lfloor k+1\rfloor})\min_{i=1}^{k} x_{\lfloor i\rfloor}
\end{aligned}$$

where (.) is a permutation that sorts vector $\boldsymbol{x}$ in increasing order $(x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)})$, $x_{(0)} = 0$, $\lfloor . \rfloor$ is a permutation that sorts kernel $\boldsymbol{\pi}$ in decreasing order $(1 = \pi_{\lfloor 1\rfloor} \geq \pi_{\lfloor 2\rfloor} \geq \cdots \geq \pi_{\lfloor N\rfloor})$ and $\pi_{\lfloor N+1\rfloor} = 0$.

Lemma 4.2 provides a relevant bases for formulating the macsum operator based aggregation given in Expressions (4) and (5).

Proposition 4.3.

$$\overline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}) = \sum_{k=1}^{N}\left(\varphi^+_{\lfloor k\rfloor} - \varphi^+_{\lfloor k+1\rfloor}\right).\max_{i=1}^{k} x_{\lfloor i\rfloor} + \sum_{k=1}^{N}\left(\varphi^-_{\lceil k\rceil} - \varphi^-_{\lceil k+1\rceil}\right).\min_{i=1}^{k} x_{\lceil i\rceil},$$

where $\lfloor . \rfloor$ is a permutation that sorts $\boldsymbol{\varphi}$ in decreasing order $(\varphi_{\lfloor 1\rfloor} \geq \varphi_{\lfloor 2\rfloor} \geq \cdots \geq \varphi_{\lfloor N\rfloor})$, with $\varphi_{\lfloor N+1\rfloor} = 0$ and $\lceil . \rceil$ is a permutation that sorts $\boldsymbol{\varphi}$ in increasing order $(\varphi_{\lceil 1\rceil} \leq \varphi_{\lceil 2\rceil} \leq \cdots \leq \varphi_{\lceil N\rceil})$ with $\varphi_{\lceil N+1\rceil} = 0$.

Proof. We have three cases:
1) $\exists i, j \in \Omega$ such that $\varphi_i > 0$ and $\varphi_j < 0$,
2) $\forall i \in \Omega$, $\varphi_i \geq 0$,
3) $\forall i \in \Omega$, $\varphi_i \leq 0$.

Case 1) $\exists i, j \in \Omega$ such that $\varphi_i > 0$ and $\varphi_j < 0$.
Thus $\min_{i\in\Omega} \varphi_i^+ = 0$, $\max_{i\in\Omega} \varphi_i^- = 0$, $\underline{\alpha} < 0$ and $\overline{\alpha} > 0$.
Let us define two maxitive kernels $\boldsymbol{\pi}^+$ and $\boldsymbol{\pi}^-$ by $\forall i \in \Omega$, $\pi_i^+ = \frac{1}{\overline{\alpha}}.\varphi_i^+$ and $\forall i \in \Omega$, $\pi_i^- = \frac{1}{\underline{\alpha}}.\varphi_i^-$.
By construction $\min_{i\in\Omega} \pi_i^- = \min_{i\in\Omega} \pi_i^+ = 0$ and $\max_{i\in\Omega} \pi_i^- = \max_{i\in\Omega} \pi_i^+ = 1$.
Moreover $\lfloor . \rfloor$ sorts $\boldsymbol{\pi}^+$ in decreasing order ($1 = \pi^+_{\lfloor 1 \rfloor} \geq \pi^+_{\lfloor 2 \rfloor} \geq \cdots \geq \pi^+_{\lfloor N \rfloor} = 0$) and $\lceil . \rceil$ sorts $\boldsymbol{\pi}^-$ in decreasing order ($1 = \pi^-_{\lceil 1 \rceil} \geq \pi^-_{\lceil 2 \rceil} \geq \cdots \geq \pi^-_{\lceil N \rceil} = 0$).
Let $(.)$ be a permutation that sorts $\boldsymbol{x}$ in increasing order ($x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$), with $x_{(0)} = 0$. Let us rewrite $\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x})$:

$$\begin{aligned}
\overline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}) &= \sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right) . \max_{i=k}^{N} \varphi^+_{(i)} + \underline{\alpha}.x_{(N)} - \sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right) . \min_{i=1}^{k-1} \varphi^-_{(i)}, \\
&= \overline{\alpha}. \sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right) . \max_{i=k}^{N} \pi^+_{(i)} + \underline{\alpha}.x_{(N)} - \underline{\alpha}. \sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right) . \max_{i=1}^{k-1} \pi^-_{(i)}, \\
&= \overline{\alpha}. \sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right) . \max_{i=k}^{N} \pi^+_{(i)} + \underline{\alpha}. \left(x_{(N)} - \sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right) . \max_{i=1}^{k-1} \pi^-_{(i)} \right).
\end{aligned}$$

Due to Lemma 4.2, we have:

$$\begin{aligned}
\sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right) . \max_{i=k}^{N} \pi^+_{(i)} &= \sum_{k=1}^{N} \left(\pi^+_{\lfloor k \rfloor} - \pi^+_{\lfloor k+1 \rfloor}\right) . \max_{i=1}^{k} x_{\lfloor i \rfloor}, \text{ and} \\
x_{(N)} - \sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right) . \max_{i=1}^{k-1} \pi^-_{(i)} &= \sum_{k=1}^{N} \left(\pi^-_{\lceil k \rceil} - \pi^-_{\lceil k+1 \rceil}\right) . \min_{i=1}^{k} x_{\lceil i \rceil}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\overline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}) &= \overline{\alpha}. \sum_{k=1}^{N} \left(\pi^+_{\lfloor k \rfloor} - \pi^+_{\lfloor k+1 \rfloor}\right) . \max_{i=1}^{k} x_{\lfloor i \rfloor} + \underline{\alpha}. \sum_{k=1}^{N} \left(\pi^-_{\lceil k \rceil} - \pi^-_{\lceil k+1 \rceil}\right) . \min_{i=1}^{k} x_{\lceil i \rceil}, \\
&= \sum_{k=1}^{N} \left(\overline{\alpha}.\pi^+_{\lfloor k \rfloor} - \overline{\alpha}.\pi^+_{\lfloor k+1 \rfloor}\right) . \max_{i=1}^{k} x_{\lfloor i \rfloor} + \sum_{k=1}^{N} \left(\underline{\alpha}.\pi^-_{\lceil k \rceil} - \underline{\alpha}.\pi^-_{\lceil k+1 \rceil}\right) . \min_{i=1}^{k} x_{\lceil i \rceil}, \\
&= \sum_{k=1}^{N} \left(\varphi^+_{\lfloor k \rfloor} - \varphi^+_{\lfloor k+1 \rfloor}\right) . \max_{i=1}^{k} x_{\lfloor i \rfloor} + \sum_{k=1}^{N} \left(\varphi^-_{\lceil k \rceil} - \varphi^-_{\lceil k+1 \rceil}\right) . \min_{i=1}^{k} x_{\lceil i \rceil}.
\end{aligned}$$

Case 2) $\forall i \in \Omega$, $\varphi_i \geq 0$.
Thus $\underline{\alpha}.x_{(N)} - \sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right).\min_{i=1}^{k-1} \varphi_{(i)}^{-} = 0$. As previously, let $\boldsymbol{\pi}^{+}$ be the maxitive kernel such that: $\forall i \in \Omega$, $\pi_i^{+} = \frac{1}{\overline{\alpha}}.\varphi_i^{+}$. Thus,

$$\overline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}) = \overline{\alpha}.\sum_{k=1}^{N} \left(\pi_{\lfloor k \rfloor}^{+} - \pi_{\lfloor k+1 \rfloor}^{+}\right).\max_{i=1}^{k} x_{\lfloor i \rfloor} = \sum_{k=1}^{N} \left(\varphi_{\lfloor k \rfloor}^{+} - \varphi_{\lfloor k+1 \rfloor}^{+}\right).\max_{i=1}^{k} x_{\lfloor i \rfloor}.$$

Case 3) $\forall i \in \Omega$, $\varphi_i \leq 0$.
Thus $\sum_{k=1}^{N} \left(x_{(k)} - x_{(k-1)}\right).\max_{i=k}^{N} \varphi_{(i)}^{+} = 0$. As previously, let $\boldsymbol{\pi}^{-}$ be the maxitive kernel such that: $\forall i \in \Omega$, $\pi_i^{-} = \frac{1}{\underline{\alpha}}.\varphi_i^{-}$. Thus,

$$\overline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}) = \underline{\alpha}.\sum_{k=1}^{N} \left(\pi_{\lceil k \rceil}^{-} - \pi_{\lceil k+1 \rceil}^{-}\right).\min_{i=1}^{k} x_{\lceil i \rceil} = \sum_{k=1}^{N} \left(\varphi_{\lceil k \rceil}^{-} - \varphi_{\lceil k+1 \rceil}^{-}\right).\min_{i=1}^{k} x_{\lceil i \rceil}.$$

□

Proposition 4.4.

$$\underline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}^{c}}(\boldsymbol{x}) = \sum_{k=1}^{N} \left(\varphi_{\lfloor k \rfloor}^{+} - \varphi_{\lfloor k+1 \rfloor}^{+}\right).\min_{i=1}^{k} x_{\lfloor i \rfloor} + \sum_{k=1}^{N} \left(\varphi_{\lceil k \rceil}^{-} - \varphi_{\lceil k+1 \rceil}^{-}\right).\max_{i=1}^{k} x_{\lceil i \rceil},$$

where $\lfloor . \rfloor$ is a permutation that sorts $\boldsymbol{\varphi}$ in decreasing order ($\varphi_{\lfloor 1 \rfloor} \geq \varphi_{\lfloor 2 \rfloor} \geq \cdots \geq \varphi_{\lfloor N \rfloor}$), with $\varphi_{\lfloor N+1 \rfloor} = \varphi_{\lfloor N \rfloor}$ and $\lceil . \rceil$ is a permutation that sorts $\boldsymbol{\varphi}$ in increasing order ($\varphi_{\lceil 1 \rceil} \leq \varphi_{\lceil 2 \rceil} \leq \cdots \leq \varphi_{\lceil N \rceil}$) with $\varphi_{\lceil N+1 \rceil} = \varphi_{\lceil N \rceil}$.

Proof. The proof is trivial by considering $\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}^{c}}(\boldsymbol{x}) = -\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(-\boldsymbol{x})$. □

4.4. The macsum aggregation can be viewed as an imprecise linear aggregation
Proposition 4.5. *Let $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ be a kernel of Ω, then $\forall \boldsymbol{x} \in \mathbb{R}^N$,*
$\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi}) \iff \mathcal{A}_{\lambda}(\boldsymbol{x}, \boldsymbol{\psi}) \in \mathcal{A}_{\nu}(\boldsymbol{x}, \boldsymbol{\varphi})$.

Proof. By construction, $\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi}) \Leftrightarrow \forall A \subseteq \Omega$, $\nu_{\boldsymbol{\varphi}}^{c}(A) \leq \lambda_{\boldsymbol{\psi}}(A) \leq \nu_{\boldsymbol{\varphi}}(A)$. Due to [34] Theorem 3, this implies that $\forall \boldsymbol{x} \in \mathbb{R}^N$, $\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}^{c}}(\boldsymbol{x}) \leq \check{\mathbb{C}}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}) \leq \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x})$. □

Proposition 4.6. *$\forall \boldsymbol{\varphi} \in \mathcal{M}(\Omega)$, $\forall \boldsymbol{\psi} \in \mathcal{M}(\Omega)$, $\lambda_{\boldsymbol{\psi}}(\Omega) = \nu_{\boldsymbol{\varphi}}(\Omega)$, i.e. $\lambda_{\boldsymbol{\psi}}$ and $\nu_{\boldsymbol{\varphi}}$ have the same gain α.*

The proof is straightforward since $\nu_{\boldsymbol{\varphi}}^{c}(\Omega) \leq \lambda_{\boldsymbol{\psi}}(\Omega) \leq \nu_{\boldsymbol{\varphi}}(\Omega)$ and $\nu_{\boldsymbol{\varphi}}^{c}(\Omega) = \nu_{\boldsymbol{\varphi}}(\Omega)$.

Proposition 4.7. *Let $\boldsymbol{\varphi} \in \mathcal{K}(\Omega)$ be a kernel of Ω, let $\boldsymbol{x} \in \mathbb{R}^N$ be a constant vector (i.e. $\forall i \in \Omega$, $x_i = c$), then $\forall \boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi}), \mathcal{A}_{\lambda}(\boldsymbol{x}, \boldsymbol{\psi}) = \mathcal{A}_{\nu}(\boldsymbol{x}, \boldsymbol{\varphi}) = \alpha.c$, with $\alpha = \sum_{i \in \Omega} \psi_i = \max_{i \in \Omega} \varphi_i^{+} + \min_{i \in \Omega} \varphi_i^{-}$.*

Proof. With $\boldsymbol{x}$ being a constant vector, it can be rewritten as $c.\boldsymbol{\chi}_{\Omega}$, where $\boldsymbol{\chi}_{\Omega}$ is the characteristic function of Ω defined by: $\forall k \in \Omega$, $\chi_{\Omega_k} = 1$. Due to the property of the Choquet integral, with μ being a set function, $\check{\mathbb{C}}_{\mu}(c.\boldsymbol{\chi}_{\Omega}) = c.\check{\mathbb{C}}_{\mu}(\boldsymbol{\chi}_{\Omega}) = c.\mu(\Omega)$.
Thus, $\mathcal{A}_{\lambda}(c.\boldsymbol{\chi}_{\Omega}, \boldsymbol{\psi}) = c.\lambda_{\boldsymbol{\psi}}(\Omega) = c.\alpha$ and $\mathcal{A}_{\nu}(c.\boldsymbol{\chi}_{\Omega}, \boldsymbol{\varphi}) = c.\left[\nu_{\boldsymbol{\varphi}}^{c}(\Omega), \nu_{\boldsymbol{\varphi}}(\Omega)\right] = c.[\alpha, \alpha] = c.\alpha$. □

Remark 1. *$\forall \boldsymbol{\psi} \in \mathcal{M}(\Omega)$ $\lambda_{\boldsymbol{\psi}}(\Omega) = \nu_{\boldsymbol{\varphi}}(\Omega)$, i.e. $\lambda_{\boldsymbol{\psi}}$ and $\nu_{\boldsymbol{\varphi}}$ have the same gain.*

Proposition 4.8. *The macsum aggregation can be viewed as an imprecise linear aggregation since,*
$\forall \boldsymbol{\varphi} \in \mathcal{K}(\Omega)$, $\forall \boldsymbol{x} \in \mathbb{R}^N$ and $\forall \beta \in \mathbb{R}$, $\mathcal{A}_\nu(\beta.\boldsymbol{x}, \boldsymbol{\varphi}) = \beta.\mathcal{A}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})$.

Proof. When $\beta \geq 0$, this property is known as *positive homogeneity* in [7].
Now note that if $\beta < 0$, then $\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\beta.\boldsymbol{x}) = \beta.\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x})$.
Thus $\mathcal{A}_\nu(\beta.\boldsymbol{x}, \boldsymbol{\varphi}) = [\beta.\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}), \beta.\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x})] = \beta.[\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x}), \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x})] = \beta.\mathcal{A}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})$. □

5. Learning a macsum relation

Let us consider a MISO system that outputs a real value $y \in \mathbb{R}$ for any input $\boldsymbol{x} \in \mathbb{R}^N$. We can assume that the link between $\boldsymbol{x}$ and y is due to an underlying unknown function g: $g(\boldsymbol{x}) = y$. Now let us suppose that we have collected a dataset of $\mathcal{X} \times \mathcal{Y} = \left\{(\boldsymbol{x}^j, y_j)\right\}_{j=1\ldots M}$ of M input-output pairs. Assuming that these M pairs are representative of the mapping $g : \mathbb{R}^N \to \mathbb{R}$, it may be possible to find a function $\hat{g}$ whose behavior is as close as possible to g. This is what is called *learning* g and the dataset $\mathcal{X} \times \mathcal{Y}$ is called *the learning dataset*. The regression approach consists of supposing that g behaves like a particular parametric function denoted $f_{\boldsymbol{\psi}}(.)$ where $\boldsymbol{\psi}$ is a set of parameters. Therefore the aim is to find $\hat{\boldsymbol{\psi}}$ such that $\forall j = 1 \ldots M$, $f_{\hat{\boldsymbol{\psi}}}(\boldsymbol{x}^j)$ is as close as possible to y_j w.r.t. a particular distance. The most commonly used distance is the quadratic norm for different reasons. The main reason in this context is that learning often involves deriving the distance and the quadratic distance is easy to derive. The quadratic distance can be expressed as $\mathbb{L}(\{y_j, f_{\boldsymbol{\psi}}(\boldsymbol{x}^j)\}_{j=1\ldots M}) = \sum_{j=1}^{M} ||y_j - f_{\boldsymbol{\psi}}(\boldsymbol{x}^j)||_2^2$. Finding parameters that minimize the quadratic distance is usually referred to as *solving the least squares problem* and denoted:

$$\hat{\boldsymbol{\psi}} = \underset{\boldsymbol{\psi}}{\operatorname{argmin}}\ \mathbb{L}(\{y_j, f_{\boldsymbol{\psi}}(\boldsymbol{x}^j)\}_{j=1\ldots M}). \tag{6}$$

5.1. Linear regression

In many cases, input-output relationship is supposed to be linear, either globally or locally. Linear systems are characterized by two main properties: homogeneity and additivity. Homogeneity means that a change in inputs results in a change in outputs – which seems to be a nice property for systems – wich can be obliterated in case of saturation (e.g. with digitization) or hysteresis (e.g. mechanical systems). Additivity can be mathematically expressed as: $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^N$, $g(\boldsymbol{x}_1 + \boldsymbol{x}_2) = g(\boldsymbol{x}_1) + g(\boldsymbol{x}_2)$. We therefore also have $\forall \boldsymbol{x} \in \mathbb{R}^N$, $\forall \beta \in \mathbb{R}$, $g(\beta.\boldsymbol{x}) = \beta.g(\boldsymbol{x})$.

A third property, i.e. shift invariance, can be mandatory for linear systems. Shift invariance means that the system characteristics do not change with time (or space). This last characteristic is rather challenging in signal processing because some hidden variables change the system behavior with time or space

(e.g. heat for electronic circuit), or because the signal has been measured by different sensors with slightly different proprieties (e.g. pixel sensors for image acquisition through a retina).

The first two properties are perfectly taken into account by modeling the system via the additive aggregation proposed in Section 4.2. The parametric function f boils down to $f_{\boldsymbol{\psi}}(.) = \mathcal{A}_\lambda(., \boldsymbol{\psi})$, where $\boldsymbol{\psi} \in \mathcal{K}(\Omega)$ is a kernel that gathers all of the N parametric values.

Considering the quadratic distance, learning a linear representation of the unknown function g based on the $\mathcal{X} \times \mathcal{Y}$ dataset consists of finding $\hat{\boldsymbol{\psi}}$ solving the least square problem expressed in Equation (6). This technique is known as *linear regression*. Such minimization can be obtained by a gradient descent, but can also be obtained directly by: $\hat{\boldsymbol{\psi}} = \boldsymbol{X}^\dagger.\boldsymbol{Y}$, where $\boldsymbol{X}$ is the matrix $\left[x_i^j\right]$ ($j = 1 \dots M$ being the row index and $i = 1 \dots N$ being the column index), $\boldsymbol{Y}$ the column matrix $[y_j]$ ($j = 1 \dots M$) and $\boldsymbol{X}^\dagger$ being the Moore-Penrose pseudo-inverse of $\boldsymbol{X}$.

In this context, the shift invariance property is associated with the fact that the value of $\hat{\boldsymbol{\psi}}$ after convergence of the estimator is unique, whatever the subset of $\mathcal{X} \times \mathcal{Y}$ considered.

On the contrary, if the shift invariance is not fulfilled, it means that there are potentially several $\hat{\boldsymbol{\psi}}$ values that can be considered for modeling the system behavior, depending on time, location or other external factors that cannot be included in the modeling. This means that the same input value can lead to different outputs, or that the same output value can be caused by different input values. This multiple mapping can be regarded as random or imprecise process. Here we propose to account for shift variance by considering a (convex) set of kernels instead of a single kernel.

5.2. Towards imprecise linear regression

Let us now suppose that the shift invariance is not fulfilled but homogeneity and linearity are fulfilled at least locally (e.g. for the same pixel, at the same temperature or during a limited time period). Let us also suppose that all possible linear representatives of the unknown system have the same gain and that the set of representatives is convex (see Section 2.3.3). This situation can be perfectly handheld by considering the macsum operator $\nu_{\boldsymbol{\varphi}}$: as shown in Section 4.4, the macsum operator can be considered as an imprecise linear operator whose elements are linear operators having the same gain.

Learning this imprecise operator thus boils down to estimating $\hat{\boldsymbol{\varphi}}$ that best represents this convex set of linear operators. Two major issues remain to be addressed.

1– What can the equivalent of least squares be in this imprecise learning context?

2– How to address the leaning phase, i.e. what is the relevant method to minimize this ad hoc new cost function?

5.2.1. The cost function

The goal of *learning a macsum relation* is to find a kernel $\hat{\boldsymbol{\varphi}} \in \mathcal{K}(\Omega)$ that ensures that the value $\mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})$ as close as possible to y_j $\forall j \in \{1, \ldots, M\}$. Since $\mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})$ is an interval, the question that should be addressed is *what does it mean for a real interval to be close to a real value?* Different extensions of distances have been proposed to answer this question that are reviewed in [29]. Most of those solutions lead to distances that are either not relevant for this problem or too non-linear to be used in an efficient iterative learning process. Here we would like to propose a simple, obvious and easy to use learning solution. Considering $\mathcal{M}(\boldsymbol{\varphi})$ as a convex set of kernels, $[\underline{y}_j, \overline{y}_j] = \mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})$ can be considered as a convex set of outputs for linear aggregation. Making this set as close as possible to y_j can be viewed as making both bounds $\underline{y}_j$ and $\overline{y}_j$ as close as possible to y_j. This leads to the extension of the quadratic distance we propose:

$$\begin{aligned}\mathbb{L}(\{y_j, [\underline{y}_j, \overline{y}_j]\}_{j=1\ldots M}) &= \sum_{j=1}^{M} ||y_j - \underline{y}_j||_2^2 + \sum_{j=1}^{M} ||y_j - \overline{y}_j||_2^2, \\ &= \sum_{j=1}^{M} \left(2.y_j^2 - 2.y_j.\overline{y}_j - 2.y_j.\underline{y}_j + \overline{y}_j{}^2 + \underline{y}_j^2\right).\end{aligned}$$

Thus learning a macsum relation based on this extension of the quadratic distance can be formulated as:

$$\hat{\boldsymbol{\varphi}} = \underset{\boldsymbol{\varphi}}{\operatorname{argmin}}\ \mathbb{L}(\{y_j, \mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})\}_{j=1\ldots M}). \tag{7}$$

5.2.2. Gradient descent

Equation (7) can be solved using a gradient descent algorithm. A gradient descent algorithm is based on finding a local minimum of a differentiable function by repeated updating steps in the direction opposite that of the gradient of the function at the current point. This can be achieved globally (for all the learning set at each iteration) or element-by-element of the $\mathcal{X} \times \mathcal{Y}$ dataset. If the global approach is chosen, the method can be summarized by the following process:

$$\boldsymbol{\varphi}^{t+1} = \boldsymbol{\varphi}^t - \beta.\nabla\mathbb{L}(\{y_j, \mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})\}_{j=1\ldots M}),$$

where $\nabla\mathbb{L}(\{y_j, \mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi}^t)\}_{j=1\ldots M})$ is the gradient of the distance $\mathbb{L}$ when the $\boldsymbol{\varphi}^t$ value is considered. $\boldsymbol{\varphi}^t$ is the value of the estimate of $\boldsymbol{\varphi}$ at iteration t and β is a positive constant value ensuring the convergence of this iterative process. Considering that the chosen quadratic distance is locally convex and with an appropriate choice of β, the sequence $\boldsymbol{\varphi}^0, \boldsymbol{\varphi}^1, \ldots, \boldsymbol{\varphi}^t, \ldots$ converges towards the least square solution. When the element-by-element solution is chosen, then the method follows the same process but we have to replace $\nabla\mathbb{L}(\{y_j, \mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})\}_{j=1\ldots M})$ by $\nabla\mathbb{L}(y_j, \mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi}))$ and repeat each step M times.

Let us now consider Δ_k, i.e. the kth element of $\nabla\mathbb{L}(\{y_j, \mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi}^t)\}_{j=1\dots M})$. We have:

$$\begin{aligned}
\Delta_k &= \frac{\delta\mathbb{L}(\{y_j, \mathcal{A}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})\}_{j=1\dots M})}{\delta\varphi_k}, \\
&= \sum_{j=1}^{M}\left(\frac{\delta\overline{\mathcal{A}}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})^2}{\delta\varphi_k} + \frac{\delta\underline{\mathcal{A}}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})^2}{\delta\varphi_k} - y_j.\frac{\delta\underline{\mathcal{A}}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})}{\delta\varphi_k} - y_j.\frac{\delta\overline{\mathcal{A}}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})}{\delta\varphi_k}\right), \\
&= \sum_{j=1}^{M}\left(2.(\underline{\mathcal{A}}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi}) - y_j).\frac{\delta\underline{\mathcal{A}}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})}{\delta\varphi_k}\right) + \sum_{j=1}^{M}\left(2.(\overline{\mathcal{A}}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi}) - y_j).\frac{\delta\overline{\mathcal{A}}_\nu(\boldsymbol{x}^j, \boldsymbol{\varphi})}{\delta\varphi_k}\right).
\end{aligned}$$

To implement this gradient descent algorithm, the upper and lower bounds of the macsum aggregation operator must be differentiated w.r.t. each component of the sought after kernel.

5.3. Differentiating the macsum aggregation

To make the derivation of the macsum aggregation w.r.t. each component of the kernel easier, we propose to rewrite $\underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi}) = \check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x})$ and $\overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi}) = \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x})$ in a simpler form:

Proposition 5.1.

$$\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}) = \sum_{k=1}^{N}\varphi^+_{\lfloor k\rfloor}.\left(\max_{i=1}^{k} x_{\lfloor i\rfloor} - \max_{i=1}^{k-1} x_{\lfloor i\rfloor}\right) + \sum_{k=1}^{N}\varphi^-_{\lceil k\rceil}.\left(\min_{i=1}^{k} x_{\lceil i\rceil} - \min_{i=1}^{k-1} x_{\lceil i\rceil}\right), \tag{8}$$

where $\lfloor . \rfloor$ is a permutation that sorts $\boldsymbol{\varphi}$ in decreasing order ($\varphi_{\lfloor 1\rfloor} \geq \varphi_{\lfloor 2\rfloor} \geq \dots \geq \varphi_{\lfloor N\rfloor}$) with $\varphi_{\lfloor N+1\rfloor} = 0$ and $\lceil . \rceil$ is a permutation that sorts $\boldsymbol{\varphi}$ in increasing order ($\varphi_{\lceil 1\rceil} \leq \varphi_{\lceil 2\rceil} \leq \dots \leq \varphi_{\lceil N\rceil}$) with $\varphi_{\lceil N+1\rceil} = 0$ and with $\max_{i=1}^{0} x_{\lfloor i\rfloor} = 0 = \min_{i=1}^{0} x_{\lceil i\rceil}$.

Proof. Equation (4.3) gives:

$$\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}) = \sum_{k=1}^{N}\left(\varphi^+_{\lfloor k\rfloor} - \varphi^+_{\lfloor k+1\rfloor}\right).\max_{i=1}^{k} x_{\lfloor i\rfloor} + \sum_{k=1}^{N}\left(\varphi^-_{\lceil k\rceil} - \varphi^-_{\lceil k+1\rceil}\right).\min_{i=1}^{k} x_{\lceil i\rceil}.$$

Let us rewrite:

$$\begin{aligned}
\sum_{k=1}^{N}\left(\varphi^+_{\lfloor k\rfloor} - \varphi^+_{\lfloor k+1\rfloor}\right).\max_{i=1}^{k} x_{\lfloor i\rfloor} &= \sum_{k=1}^{N}\varphi^+_{\lfloor k\rfloor}\max_{i=1}^{k} x_{\lfloor i\rfloor} - \sum_{k=1}^{N}\varphi^+_{\lfloor k+1\rfloor}\max_{i=1}^{k} x_{\lfloor i\rfloor} \\
&= \sum_{k=1}^{N}\varphi^+_{\lfloor k\rfloor}\max_{i=1}^{k} x_{\lfloor i\rfloor} - \sum_{k=2}^{N+1}\varphi^+_{\lfloor k\rfloor}\max_{i=1}^{k-1} x_{\lfloor i\rfloor} \\
&= \sum_{k=1}^{N}\varphi^+_{\lfloor k\rfloor}\max_{i=1}^{k} x_{\lfloor i\rfloor} - \sum_{k=1}^{N}\varphi^+_{\lfloor k\rfloor}\max_{i=1}^{k-1} x_{\lfloor i\rfloor} - \varphi^+_{\lfloor N+1\rfloor}.\max_{i=1}^{N} x_{\lfloor i\rfloor} + \varphi^+_{\lfloor 1\rfloor}.\max_{i=1}^{0} x_{\lfloor i\rfloor} \\
&= \sum_{k=1}^{N}\varphi^+_{\lfloor k\rfloor}.\left(\max_{i=1}^{k} x_{\lfloor i\rfloor} - \max_{i=1}^{k-1} x_{\lfloor i\rfloor}\right),
\end{aligned}$$

since $\varphi^+_{\lfloor N+1 \rfloor} = 0$, and

$$\begin{aligned}
\sum_{k=1}^{N} \left(\varphi^-_{\lceil k \rceil} - \varphi^-_{\lceil k+1 \rceil}\right) . \min_{i=1}^{k} x_{\lceil i \rceil} &= \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} \min_{i=1}^{k} x_{\lceil i \rceil} - \sum_{k=1}^{N} \varphi^-_{\lceil k+1 \rceil} \min_{i=1}^{k} x_{\lceil i \rceil} \\
&= \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} \min_{i=1}^{k} x_{\lceil i \rceil} - \sum_{k=2}^{N+1} \varphi^-_{\lceil k \rceil} . \min_{i=1}^{k-1} x_{\lceil i \rceil} \\
&= \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} \min_{i=1}^{k} x_{\lceil i \rceil} - \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} \min_{i=1}^{k-1} x_{\lceil i \rceil} - \varphi^-_{\lceil N+1 \rceil} . \min_{i=1}^{N} x_{\lceil i \rceil} + \varphi^-_{\lceil 1 \rceil} . \min_{i=1}^{0} x_{\lceil i \rceil} \\
&= \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} . \left(\min_{i=1}^{k} x_{\lceil i \rceil} - \min_{i=1}^{k-1} x_{\lceil i \rceil}\right),
\end{aligned}$$

since $\varphi^-_{\lceil N+1 \rceil} = 0$.

Thus,

$$\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x}) = \sum_{k=1}^{N} \varphi^+_{\lfloor k \rfloor} . \left(\max_{i=1}^{k} x_{\lfloor i \rfloor} - \max_{i=1}^{k-1} x_{\lfloor i \rfloor}\right) + \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} . \left(\min_{i=1}^{k} x_{\lceil i \rceil} - \min_{i=1}^{k-1} x_{\lceil i \rceil}\right).$$

□

Proposition 5.2.

$$\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x}) = \sum_{k=1}^{N} \varphi^+_{\lfloor k \rfloor} . \left(\min_{i=1}^{k} x_{\lfloor i \rfloor} - \min_{i=1}^{k-1} x_{\lfloor i \rfloor}\right) + \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} . \left(\max_{i=1}^{k} x_{\lceil i \rceil} - \max_{i=1}^{k-1} x_{\lceil i \rceil}\right), \tag{9}$$

with $\min_{i=1}^{0} x_{\lfloor i \rfloor} = 0 = \max_{i=1}^{0} x_{\lceil i \rceil}$.

Proof. Proving Proposition 5.2 is straighforward by considering the equality: $\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x}) = -\check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(-\boldsymbol{x})$ and the fact that sorting $\boldsymbol{x}$ in ascending order is equivalent to sorting $-\boldsymbol{x}$ in descending order. □

Proposition 5.3. *$\forall k \in \{1, ..., N\}$, let be l the index such that $\lfloor l \rfloor = k$ and u the index such that $\lceil u \rceil = k$, then:*

$$\frac{\delta \overline{\mathcal{A}}_{\nu}(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k} = \left(\max_{i=1}^{l} x_{\lfloor i \rfloor} - \max_{i=1}^{l-1} x_{\lfloor i \rfloor}\right) + \left(\min_{i=1}^{u} x_{\lceil i \rceil} - \min_{i=1}^{u-1} x_{\lceil i \rceil}\right),$$

and

$$\frac{\delta \underline{\mathcal{A}}_{\nu}(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k} = \left(\min_{i=1}^{l} x_{\lfloor i \rfloor} - \min_{i=1}^{l-1} x_{\lfloor i \rfloor}\right) + \left(\max_{i=1}^{u} x_{\lceil i \rceil} - \max_{i=1}^{u-1} x_{\lceil i \rceil}\right).$$

with $\min_{i=1}^{0} x_{\lfloor i \rfloor} = 0 = \max_{i=1}^{0} x_{\lceil i \rceil}$.

Proof. First remember that $\forall k \in \Omega$, $\varphi_k^+ = \max(\varphi_k, 0)$ and $\varphi_k^- = \min(\varphi_k, 0)$ and thus $\varphi_k = \varphi_k^+ + \varphi_k^-$ and $\delta\varphi_k = \delta\varphi_k^+ + \delta\varphi_k^-$. Therefore,

$$\frac{\delta \underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k} = \frac{\delta \underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k^+} . \frac{\delta \varphi_k^+}{\delta \varphi_k} + \frac{\delta \underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k^-} . \frac{\delta \varphi_k^-}{\delta \varphi_k},$$
$$\frac{\delta \overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k} = \frac{\delta \overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k^+} . \frac{\delta \varphi_k^+}{\delta \varphi_k} + \frac{\delta \overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k^-} . \frac{\delta \varphi_k^-}{\delta \varphi_k}.$$

Considering $\frac{\delta \varphi_k}{\delta \varphi_k^+} = \frac{\delta \varphi_k^+}{\delta \varphi_k^+} + \frac{\delta \varphi_k^-}{\delta \varphi_k^+} = 1$ as well as $\frac{\delta \varphi_k}{\delta \varphi_k^-} = \frac{\delta \varphi_k^+}{\delta \varphi_k^-} + \frac{\delta \varphi_k^-}{\delta \varphi_k^-} = 1$, thus:

$$\frac{\delta \underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k} = \frac{\delta \underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k^+} + \frac{\delta \underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k^-} \text{ and } \frac{\delta \overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k} = \frac{\delta \overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k^+} + \frac{\delta \overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k^-}.$$

Now, proving Proposition 5.3 is straighforward by differentiating Equations (8) and (9) w.r.t. φ_k^+ and φ_k^-.

□

Let $\underline{\boldsymbol{\delta y}} \in \mathbb{R}^N$ be the gradient of $\underline{\boldsymbol{y}}$ w.r.t. $\boldsymbol{\varphi}$ and $\overline{\boldsymbol{\delta y}} \in \mathbb{R}^N$ be the gradient of $\overline{\boldsymbol{y}}$ w.r.t. $\boldsymbol{\varphi}$: $\forall k \in \Omega$, $\underline{\delta y}_k = \frac{\delta \underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k}$ and $\overline{\delta y}_k = \frac{\delta \overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k}$, computing the derivatives of both $\underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})$ and $\overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})$ can easily be achieved using Algorithm 1.

6. Experiments

The experiment we describe, in the image processing context, aims to highlight the properties of both the macsum aggregation operator and our proposed learning process.

Most image processing techniques suppose that the point spread function (PSF) of each sensor that composes a camera retina (also called physical pixels) is space translation invariant. Hence, each element of the retina is supposed to similarly measure the amount of light that irradiates it. However, this translation invariance is usually only approximate. By construction, the PSF of each pixel can have fluctuations [20], i.e. the way each retina element integrates the irradiance can vary. This variation is usually unnoticeable to the user since prior retina calibration avoids fluctuation of the measured intensities (by moving the camera, the variation cannot be perceived). However, this can have highly impact the deconvolution or image reconstruction (super-resolution, tomography, etc.) processes [4, 27]. In addition, lenses often cause unintended and undesired vignetting effects whereby the image is sharper at the center than on the edges. This phenomenon can be characterized by a pixel PSF whose specificity decreases with the distance to the center of the image. This is very noticeable in plenoptic or omnidirectional imaging and optical microscopy. Figure (1) presents an example of such a phenomenon.

Algorithm 1: Computation of $\underline{\boldsymbol{\delta y}}$, $\overline{\boldsymbol{\delta y}}$

Input: $\boldsymbol{x} = \{x_i\}_{i=1\ldots N}$, $\boldsymbol{\varphi} = \{\varphi_i\}_{i=1\ldots N}$
Output: $\underline{\boldsymbol{\delta y}}$, $\overline{\boldsymbol{\delta y}}$
set $\boldsymbol{\iota} = \{1, \ldots, N\}$;
sort $(\boldsymbol{\varphi}, \boldsymbol{\iota})$ w.r.t. $\boldsymbol{\varphi}$ in increasing order ;
$\underline{x}_1 = x_{\iota_1}$, $\overline{x}_1 = x_{\iota_1}$;
for $k = 2 \ldots N$ **do**
  $\underline{x}_k = \min(\underline{x}_{k-1}, x_{\iota_k})$;
  $\overline{x}_k = \max(\overline{x}_{k-1}, x_{\iota_k})$;
for $k = 1 \ldots N$ **do**
  $\underline{\delta y}_{\iota_k} = \underline{x}_k$;
  $\overline{\delta y}_{\iota_k} = \overline{x}_k$;
for $k = 2 \ldots N$ **do**
  $\underline{\delta y}_{\iota_k} = \underline{\delta y}_{\iota_k} - \underline{x}_{k-1}$;
  $\overline{\delta y}_{\iota_k} = \overline{\delta y}_{\iota_k} - \overline{x}_{k-1}$;
reverse the order of the elements of $\boldsymbol{\iota}$;
$\underline{x}_1 = x_{\iota_1}$, $\overline{x}_1 = x_{\iota_1}$;
for $k = 2 \ldots N$ **do**
  $\underline{x}_k = \min(\underline{x}_{k-1}, x_{\iota_k})$;
  $\overline{x}_k = \max(\overline{x}_{k-1}, x_{\iota_k})$;
for $k = 1 \ldots N$ **do**
  $\underline{\delta y}_{\iota_k} = \underline{\delta y}_{\iota_k} + \underline{x}_k$;
  $\overline{\delta y}_{\iota_k} = \overline{\delta y}_{\iota_k} + \overline{x}_k$;
for $k = 2 \ldots N$ **do**
  $\underline{\delta y}_{\iota_k} = \underline{\delta y}_{\iota_k} - \underline{x}_{k-1}$;
  $\overline{\delta y}_{\iota_k} = \overline{\delta y}_{\iota_k} - \overline{x}_{k-1}$;

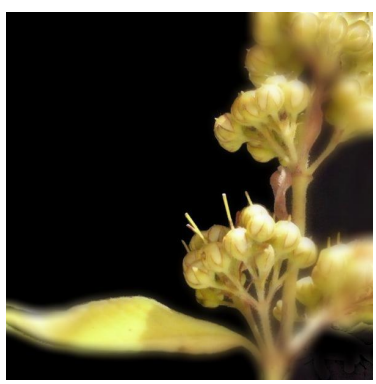

Figure 1: Image acquired by a non-shift invariant sensor.

Figure 2: Four out of the 2000 images used for this experiment.

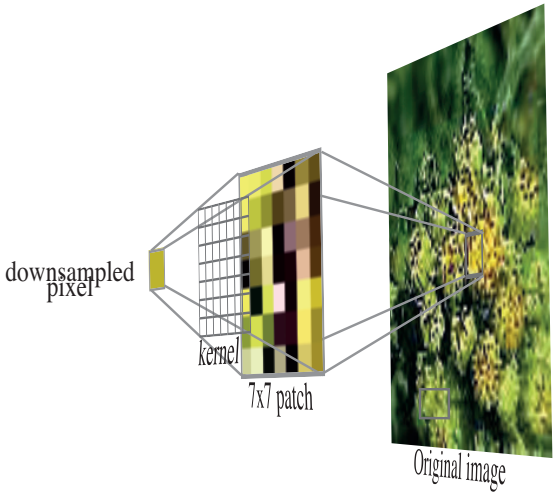

Figure 3: Downsampling the 3rd image of Figure (2)

This experiment aims to mimick a vignetting effect. We used 1000 600×600 natural images sourced from the CLEF[3] project (see Figure (2)). Each of the 1000 images was downsampled 7-fold, i.e. a 85×85 image was associated with each 600×600 imag (see Figure (5)). The value of each pixel of the 85×85 image is obtained by aggregating a 7×7 patch of the 600×600 image. This aggregation is performed by convolution with a 7×7 kernel that is supposed to be the PSF of the pixel under consideration (see Figure (3)). The specificity of the PSF associated with each pixel of the 85×85 image decreases with the distance of the pixel to the center of the image. Figure (4) plots the two extreme values of the downsampling kernel: Figure (4).a the PSF used for central values, Figure (4).b the PSF used for border values.

In compliance with the theory we promote, every PSF is signed (i.e. has positive and negative values) and integrated to 1.9.

[3]https://www.imageclef.org/

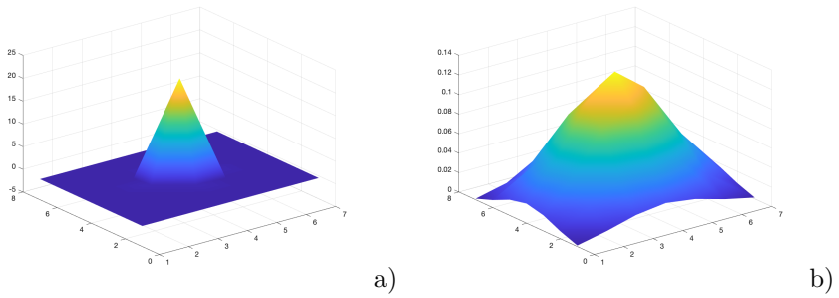

Figure 4: PSF used for image downsampling: central (a) and border (b).

Figure 5: Images depicted in Figure (2) after downsampling

In this experiment, we propose to learn the kernel associated with downsampling within both additive and macsum aggregation modeling. Downsampling with the additive model consists of computing the mean of each patch of the original image weighted by a kernel $\boldsymbol{\psi}$ (referred to here as precise downsampling). Downsampling with the macsum model consists of computing the interval-valued aggregation of each patch of the original image weighted by a kernel $\boldsymbol{\varphi}$ (referred to here as imprecise downsampling).

We use 30 randomly chosen images to learn the kernel and the 970 remaining images to characterize the result. The idea is to evaluate the extent to which this imprecise model is able to predict the closeness of the model to the truth as well as how the learning process is able to come up with a kernel that, when associated with the macsum operator, is able to represent the set of kernels used to downsample the original images.

We arbitrarily performed 1000 iterations of the learning algorithm for both models (additive and macsum). Let $\hat{\boldsymbol{\psi}}$ be the additive and $\hat{\boldsymbol{\varphi}}$ be the macsum kernel obtained by the learning algorithms (see Figure (6)). Let $\{I_k\}_{k=1\ldots1000}$ be the set of the 1000 600×600 original images, $\{F_k\}_{k=1\ldots1000}$ be the set of the 1000 85×85 downsampled images, $\{\hat{F}_k\}_{k=1\ldots1000}$ be the set of the images obtained by downsampling the I_k with $\hat{\boldsymbol{\psi}}$ and $\{[\underline{F}_k, \overline{F}_k]\}_{k=1\ldots1000}$ be the interval valued images obtained by using the macsum downsampling with the kernel $\hat{\boldsymbol{\varphi}}$.

To evaluate the closeness of the kth predicted precise downsampled image

$\hat{F}_k$ to the *true* downsampled image F_k, we use the mean L_1 distance, over the 970 test images, as defined by:
$L_1(\hat{F}_k, F_k) = \frac{1}{7225} \sum_{i=1}^{85} \sum_{j=1}^{85} |\hat{F}_k(i,j) - F_k(i,j)|$.

To evaluate the closeness of the kth predicted imprecise downsampled image $[\underline{F}_k, \overline{F}_k]$ to the *true* downsampled image F_k, we use the mean, over the 970 test images, of four extensions of the L_1 distance:

- the Hausdorff extension defined by: $L_1^H([\underline{F}_k, \overline{F}_k], F_k) = \sup_{G \in [\underline{F}_k, \overline{F}_k]} L_1(G, F_k)$,
- the natural extension defined by: $L_1^N([\underline{F}_k, \overline{F}_k], F_k) = \inf_{G \in [\underline{F}_k, \overline{F}_k]} L_1(G, F_k)$,
- the mean distance to the center defined by:
  $L_1^C([\underline{F}_k, \overline{F}_k], F_k) = L_1(\frac{1}{2}(\underline{F}_k + \overline{F}_k), F_k)$,
- and the Saulnier extension proposed in [29] denoted $L_1^S([\underline{F}_k, \overline{F}_k], F_k)$.

The Saulnier extension can be expressed as:

$$L_1^S([\underline{F}_k, \overline{F}_k], F_k) = \frac{1}{7225^2} \sum_{i=1}^{85} \sum_{j=1}^{85} \sum_{i'=1}^{85} \sum_{j'=1}^{85} |\tilde{F}_k(i,j) - F_k(i,j)| - \Delta F_k(i,j) + \Delta F_k(i',j'),$$

with $\tilde{F}_k(i,j) = \frac{1}{2}.\big(\overline{F}_k(i,j) + \underline{F}_k(i,j)\big)$ and $\Delta F_k(i,j) = \frac{1}{2}.\big(\overline{F}_k(i,j) - \underline{F}_k(i,j)\big)$. The purpose of this distance is to integrate, in a single value, both the distance of a precise vector to the center of the interval-valued vector (like the mean distance to the center) as well as the predictive power of the interval width, i.e. the correlation between the distance to the center of the interval and the width of the interval.

Those distances and extensions can be used to characterize the convergence of the learning algorithms. As can be seen in Figure (7), in both experiments (precise and imprecise learning) the convergence seems very fast (< 30 iterations for the additive and macsum learning). We can also consider the convergence of the macsum learning through the extended distances – Hausdorff, Saulnier and natural extensions – (see Figure (8)). In that case, the convergence is instead obtained after around 150 iterations. How fast the algorithms converge naturally depends on the images used for the learning process. It was never more than 60 iterations in the multiple trials we performed, where the learning image dataset was modified.

Here it is of particular interest to assess the ability of the learned macsum aggregation to predict its own output error.

First question: *Does kernel $\hat{\boldsymbol{\varphi}}$ dominate all the kernels used to downsample the image?*. To answer this question, we used a MonteCarlo process that consists of randomly selecting one kernel $\boldsymbol{\psi}$ among those used for the downsampling and one 7×7 set A and testing whether $\lambda_{\boldsymbol{\psi}}(A) \in [\nu^c_{\hat{\boldsymbol{\varphi}}}(A), \nu_{\hat{\boldsymbol{\varphi}}}(A)]$ or not. We performed this procedure 100000 times and 100% of the tests were positive. This rate also equals 100% if we consider the learned additive kernel $\hat{\boldsymbol{\psi}}$ (i.e. test whether $\lambda_{\hat{\boldsymbol{\psi}}}(A) \in [\nu^c_{\hat{\boldsymbol{\varphi}}}(A), \nu_{\hat{\boldsymbol{\varphi}}}(A)]$ or not) when repeating the additive and

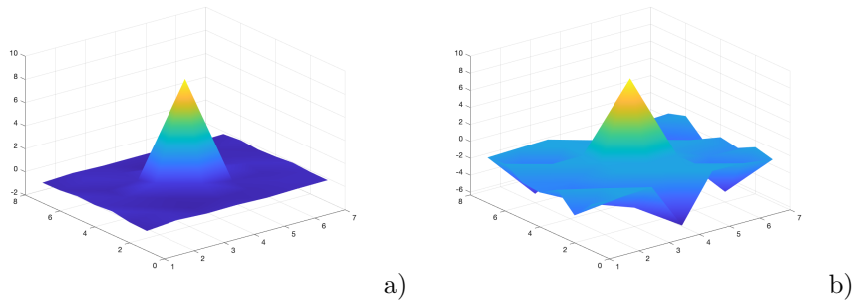

Figure 6: After the learning process, the kernel $\hat{\boldsymbol{\psi}}$ (a) and the kernel $\hat{\boldsymbol{\varphi}}$ (b)

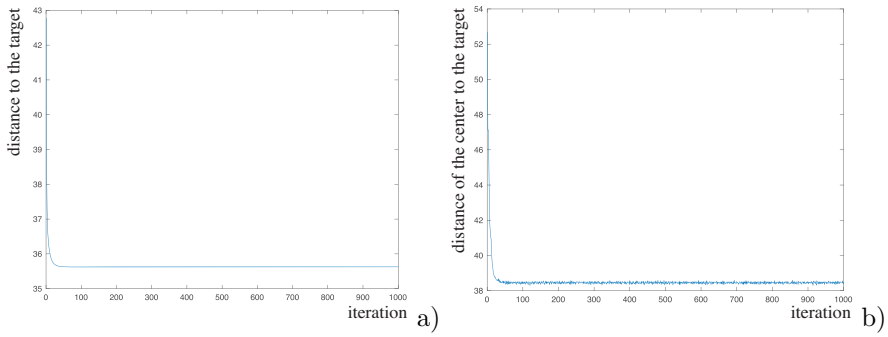

Figure 7: Convergence of the additive learning (a) and the macsum learning (b) processes

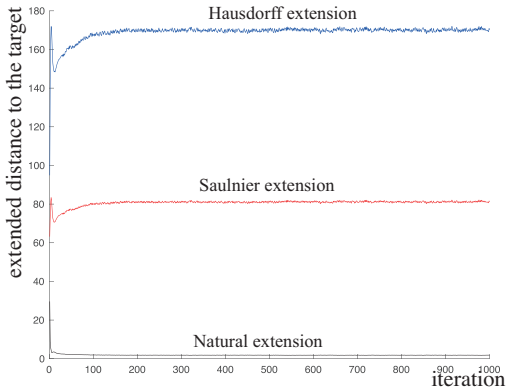

Figure 8: Convergence of the macsum learning process through extended distances

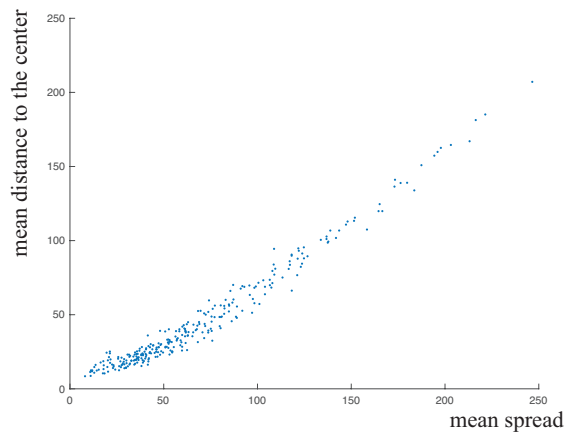

Figure 9: Correlation between the mean spread and the mean distance to the center

macsum learning processes 300 times.

Second question: *Are the downsampled images F_k included in the imprecise images $[\underline{F}_k, \overline{F}_k]$?*. To answer this question, we counted the number of pixels (i, j) of each test image, such that $F_k(i, j) \in [\underline{F}_k(i, j), \overline{F}_k(i, j)]$. We obtained a rate of 91% of pixels of the downsampled images that were in the range of the predicted imprecise values of the corresponding downsampled image.

Third question: *Is the imprecision of the imprecise predicted downsampled image a marker of the ability of the macsum operator associated wit the learned kernel $\hat{\boldsymbol{\varphi}}$ to correctly predict the output?*. To answer this third question, we computed, for each of the 1970 test images, the mean distance to the center: $\epsilon_k = L_1^C([\underline{F}_k, \overline{F}_k], F_k)$ and the mean spread of the interval-valued image $\delta_k = \frac{1}{7225} \sum_{i=1}^{85} \sum_{j=1}^{85} \frac{1}{2}(\overline{F}_k(i, j) - \underline{F}_k(i, j))$. As can be noted in Figure (9), those two values are highly correlated. The Pearson correlation coefficient between ϵ and δ is 0.98. The Pearson correlation coefficient between δ and all the other distances ranges from 0.98 to 0.99. This is also illustrated in Figure (10) where there is clearly a high resemblance between the image of the mean spread in Figure (10)(a) and the image of the distance to the center in Figure (10)(b). The error quantification ability shown in [24] with maxitive-kernel operator based aggregation seems to apply to the macsum operator based aggregation.

Finally, it would be of interest to determine the closeness of the predicted downsampled images to the true downsampled images. When considering the precise downsampling by using $\lambda_{\hat{\boldsymbol{\psi}}}$ as the aggregation operator, the mean L_1 distance computed on all the test images is 41.6. When considering the imprecise downsampling by using $\nu_{\hat{\boldsymbol{\varphi}}}$ as the aggregation operator, the mean L_1^H value is

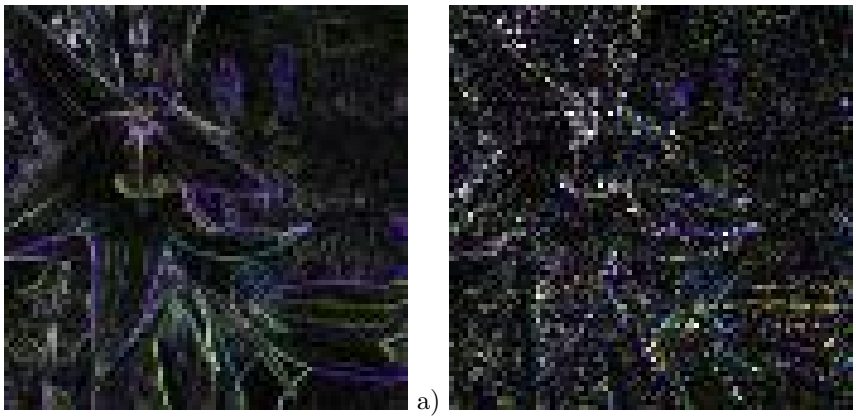

Figure 10: Image of the spread (a) and the distance to the center (b) for the 2nd image of Figure (5)

183.7, the mean L_1^S is 77.4, the mean L_1^C is 46.2 and the mean L_1^N is 2.4.

7. Conclusion

Linear regression is a fairly convenient method to use to learn parameters of an aggregation function when the system under consideration is consistent. When this consistency is not satisfied, e.g. because of translation invariance, it could be interesting to replace an aggregation based on a function by an aggregation based on a coherent set of functions. Here we proposed to represent a system by an aggregation function based on the macsum operator and to perform a linear regression in order to find the optimal parameter vector in the sense of a quadratic distance. This technique aims at learning the model of an approximately linear system whose gain is known to be invariant but whose behavior varies by translation (temporal, spatial, etc). The experiment proposed in Section 6 illustrates the validity of such an approach in this context.

This is very preliminary work. Indeed, very few authors have proposed to learn an aggregation relation that uses a Choquet integral, and even fewer if we consider non-monotonic and non-normalized set functions.

A follow-up of this work could lead to multiple possible tracks, some being applicative – finding areas in which the macsum aggregation function could be used – others theoretical – determining wether, as in the linear case, the knowledge of the kernel representing a system shed greater light on the functioning of the system. Our regression approach is based on a simple quadratic distance between the bounds of the predicted interval and the value of the output of the learning base. It could be interesting to minimize a distance more in line with the nature of the problem, i.e. an extension of the quadratic distance or of the L1 distance, such as the Saulnier extension. It could also be relevant to consider the imprecision of the measurements of both inputs and outputs of a

real system by using a pseudo-distance between intervals, for the learning process, and a macsum model whose input vector would be intervallistic. Finally, it would be interesting to see if it would not be advantageous to consider other set functions or other integrals such as generalized forms of the Choquet [36] or Sugeno integrals [9].

References

[1] L. Billard and E. Diday. Regression analysis for interval-valued data. In H. Kiers, J-P. Rasson, P. Groenen, and M. Schader, editors, *Data Analysis, Classification, and Related Methods*, pages 369–374. Springer Berlin Heidelberg, 2000.

[2] A. Bisserier, R. Boukezzoula, and S. Galichet. A revisited approach to linear fuzzy regression using trapezoidal fuzzy intervals. *Information Sciences*, 180(19):3653–3673, 2010.

[3] A. De Carvalho, Francisco T., N. Lima, A. Eufrasio, and P. Tenorio. A new method to fit a linear regression model for interval-valued data. In S. Biundo, T. Frühwirth, and G. Palm, editors, *KI 2004: Advances in Artificial Intelligence*, pages 295–306. Springer Berlin Heidelberg, 2004.

[4] M. Delbracio. *Two problems of digital image formation: recovering the camera point spread function and boosting stochastic renderers by auto-similarity filtering*. Phd thesis, École normale supérieure de Cachan, March 2013.

[5] P. Diamond. Fuzzy least squares. *Information Sciences*, 46(3):141–157, 1988.

[6] P. Diamond and H. Tanaka. *Fuzzy regression analysis*, pages 349–387. Springer US, Boston, MA, 1998.

[7] S. Doria, R. Mesiar, and A. Šeliga. Sub-additive aggregation functions and their applications in construction of coherent upper previsions. *Mathematics*, 9(1), 2021.

[8] D. Dubois and H. Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Springer, 1988.

[9] D. Dubois, H. Prade, A. Rico, and B. Teheux. Generalized qualitative sugeno integrals. *Information Sciences*, 415-416:429–445, 2017.

[10] D. Dubois and A. Rico. Axiomatisation of discrete fuzzy integrals with respect to possibility and necessity measures. In V. Torra, Y. Narukawa, G. Navarro-Arribas, and C. Yañez, editors, *Modeling Decisions for Artificial Intelligence*, pages 94–106. Springer International Publishing, 2016.

[11] B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science*, pages 54–75, 1986.

[12] A. Fallah Tehrani. Heuristics-based learning approach for choquistic regression models. *Pattern Recognition Letters*, 149:137–142, 2021.

[13] A. Fallah Tehrani, W. Cheng, K. Dembczy, and E. Hüllermeier. Learning monotone nonlinear models using the choquet integral. In *Machine Learning and Knowledge Discovery in Databases - European Conference*, pages 414–429, 08 2011.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.

[15] T. Havens and D. Anderson. Machine learning of choquet integral regression with respect to a bounded capacity (or non-monotonic fuzzy measure). In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2019.

[16] M. Hladík and M. Černý. Interval regression by tolerance analysis approach. *Fuzzy Sets and Systems*, 193:85–107, 2012.

[17] M. Hojati, C. Bector, and K. Smimou. A simple method for computation of fuzzy linear regression. *European Journal of Operational Research*, 166(1):172–184, 2005.

[18] C-H. Huang and H-Y. Kao. Interval regression analysis with soft-margin reduced support vector machine. In B-C. Chien, T-P. Hong, S-M. Chen, and M. Ali, editors, *Next-Generation Applied Intelligence*, pages 826–835. Springer Berlin Heidelberg, 2009.

[19] M. Inuiguchi and T. Tanino. Interval linear regression analysis based on minkowski difference – a bridge between traditional and interval linear regression models. *Kybernetika*, 42(4):423–440, 2006.

[20] H. Jiang, Y. Wang, X. Li, H. Zhao, and Y. Li. Space-variant point spread function measurement and interpolation at any depth based on single-pixel imaging. *Optic Express*, 28(7):9244–9258, 2020.

[21] H. Kashima, K. Yamasaki, A. Inokuchi, and H. Saigo. Regression with interval output values. In *Proceedings - International Conference on Pattern Recognition*, 2008.

[22] M. Kieffer, L. Jaulin, and E. Walter. Guaranteed recursive nonlinear state estimation using interval analysis. *Conference in decision and control*, 37(19):3966–3971, 1998.

[23] Feng Li, Shoumei Li, Nana Tang, and T. Denœux. Constrained interval-valued linear regression model. In *20th International Conference on Information Fusion*, pages 1–8, 2017.

[24] K. Loquin, O. Strauss, and J.F. Crouzet. Possibilistic signal processing: How to handle noise? *International Journal of Approximate Reasoning*, 51(9):1129–1144, 2010.

[25] M. Sugeno M. Grabisch and T. Murofushi. Fuzzy measures and integrals: theory and applications. *Heidelberg: Physica*, 2000.

[26] E. Nasrabadi and S. Hashemi. Robust fuzzy regression analysis using neural networks. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 16(04):579–598, 2008.

[27] K. Novak and A. Watnik. Imaging through deconvolution with a spatially variant point spread function. In L. Tian, J. Petruccelli, and C. Preza, editors, *Computational Imaging VI*, volume 11731, pages 24 – 40. International Society for Optics and Photonics, SPIE, 2021.

[28] G. Peters. Fuzzy linear regression with fuzzy intervals. *Fuzzy Sets and Systems*, 63(1):45–55, 1994.

[29] H. Saulnier, O. Strauss, and I. Couso. Comparing interval-valued estimations with point-valued estimations. In *16th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 595–604, 2016.

[30] S. Stoeva and A. Nikov. A fuzzy backpropagation algorithm. *Fuzzy Sets and Systems*, 112(1):27–39, 2000.

[31] O Strauss, A. Rico, and Y. Hmidy. Macsum: a new interval-valued linear operator. *International journal of approximate reasoning*, to appear.

[32] H. Tanaka. Fuzzy data analysis by possibilistic linear models. *Fuzzy Sets and Systems*, 24(3):363–375, 1987.

[33] H. Tanaka and H. Lee. Interval regression analysis by quadratic programming approach. *IEEE Transactions on Fuzzy Systems*, 6(4):473–481, 1998.

[34] A. De Waegenaere and P. Wakker. Choquet integrals with respect to non-monotonic set functions. Discussion Paper 1997-44, Tilburg University, Center for Economic Research, 1997.

[35] X. Yan and X. Su. *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co., Inc., USA, 2009.

[36] D. Zhang, R. Mesiar, and E. Pap. Pseudo-integral and generalized choquet integral. *Fuzzy Sets and Systems*, 2020.