Macsum agregation learning and missing values

Strauss Olivier[1] and Agnès Rico[2]

[1] LIRMM, Université de Montpellier, CNRS, France
[2] ERIC, Université Claude Bernard Lyon 1, CNRS, France
Olivier.Strauss@lirmm.fr Agnes.Rico@univ-lyon1.fr

Abstract. In recent work, a new kind of aggregation method has been proposed under the name of MacSum aggregation function that can be viewed as an interval valued aggregation function that is controlled by a precise vector of weights. This aggregation can be seen as a real valued extension of the possibility based aggregation. In this article, we show that a MacSum aggregation can be learned by using an input-output database where some input vectors have missing values.

Keywords: Interval valued aggregation · Choquet integral · Non monotonic set functions · Missing values · Image processing.

1 Introduction

In many applications (chemistry, medicine, robotics, economy, control, etc.) it is crucial to model the relationship between inputs and outputs of a physical process. We're interested here in multi-input, single-output processes (MISO), i.e. having a real-valued input vector and a real single valued output. In this context, linear models are widely used due to their ease of implementation and excellent predictive power. A linear model can be seen as an aggregation function of inputs producing an output that is nothing more than a weighted sum of the inputs. Aggregation is then entirely defined by the weights used. If all the weights are positive and sum to one, then a linear aggregation can be seen as a mathematical expectation based on a discrete probability distribution formed by the aggregation weights. From now on, we'll call the vector of weights used in a linear aggregation a *kernel*. A linear aggregation is entirely defined by its kernel

The linear model – i.e. its kernel – is very simple to learn from a set of inputs-outputs of the physical process to be modeled. To achieve this, more or less sophisticated regression methods are used, with the aim of bringing the model closer to the real process - at least on the training data, the most commonly used method being linear regression based on Euclidean distance.

Naturally, linear models are used to model non-linear processes, with good performance. It's no coincidence that these models are one of the key features of convolutional neural networks, which are currently revolutionizing modeling approaches. However, one of the weaknesses of linear models is the difficulty of

accessing a measure of accuracy with which the physical process is modeled, or to account for missing values in the train database.

In the 2000s, Loquin et al. proposed to build on the close relationship between probability and possibility to extend the notion of linear aggregation [5]. They propose a kind of imprecise linear aggregation governed, like ordinary linear aggregation, by a vector of weights, called maxitive kernel, of dimension equal to the dimension of the inputs. In this extension, an imprecise linear model can be considered as a convex set of precise linear models. The set of kernels that are represented by the maxitive kernel are said to be *dominated* by the maxitive kernel. As shown in a number of articles (see e.g. [4]), this modeling approach makes it very easy to take into account the imperfection of modeling a process by a linear model, while retaining the same algorithmic simplicity. Its computation is based on asymmetric Choquet integral [1]. However, a shortcoming of this extension is that it can only model sets of linear aggregation functions whose weights are positive and sum to one.

In a recent paper [7], the work of Loquin et al. has been extended to any set of weights. Under the name of MacSum, we proposed an imprecise linear aggregation operator ruled by a single kernel whose dimension equal the dimension of the input vector. MacSum aggregation takes as input two real vectors: an input vector and a kernel. Its output is a real interval corresponding to the convex set of real outputs that would have been obtained by a convex set of linear aggregations with the same gain (the gain of a linear operator is the sum of its weights).

The kernel of a MacSum aggregation can be learned from a set of inputs-outputs as in the case of a classical linear aggregation [7]. Moreover, it can take as input an interval-valued vector in order to take into account the imprecise nature of the input data (e.g. sensor data whose precision has been calibrated). This extension to imprecise inputs is achieved without any significant increase in algorithmic complexity [3].

In this article, we investigate the possibility of using the intervallist nature of the inputs to learn a MacSum model with input vectors of which some values are missing. The problem of missing values in learning is a fairly central one, to which we give an interesting answer here.

Indeed, most of the time, when some input values are missing, the range of their possible values is generally known – the range $[0, 255]$ for 8-bit quantized values, the range $[0, 5]$V for a measurement voltage, the range $[0.5, 1.5]$ g/l for fasting blood glucose, etc. In this article, we propose to replace missing values by their possible range of variation. We illustrate this proposal with an experiment in image processing.

This article is organized as follows. Section 2 presents some useful notations and definitions. Section 3 presents the MacSum aggregation model and how it can be learnt with a dataset having some missing values. Section 4 is dedicated to an illustrative experiment. We then conclude.

2 Preliminaries

In this section, we try to summarize the main points of three previous articles, namely [7], [2] and [3].

2.1 Notations

- $\Omega = \{1, \ldots, N\} \subset \mathbb{N}$ is a finite set.
- A real vector of $\mathbb{R}^N$ will be denoted $\boldsymbol{x} = (x_1, \cdots, x_N) \in \mathbb{R}^N$.
- Let $\boldsymbol{x} \in \mathbb{R}^{\mathbb{N}}$, we define $\boldsymbol{x}^+, \boldsymbol{x}^- \in \mathbb{R}^{\mathbb{N}}$ such that $\forall i \in \Omega$, $x_i^+ = \max(0, x_i)$ and $x_i^- = \min(0, x_i)$.
- $\overline{\underline{x}} = [\underline{x}, \overline{x}]$ is a real interval whose lower bound is $\underline{x}$ and upper bound is $\overline{x}$.
- $\mathbb{IR}$ is the set of real intervals.
- A vector of real intervals is an element of $\mathbb{IR}^N$ denoted $\overline{\underline{\boldsymbol{x}}} = (\overline{\underline{x}}_1, \overline{\underline{x}}_2, .., \overline{\underline{x}}_N)$.

2.2 Definitions

Let us recall briefly some definitions.

- A set function is a function $\mu : 2^\Omega \to \mathbb{R}$ that maps any subset of Ω onto a real values complying with $\mu(\varnothing) = 0$. To a set function μ is associated a complementary set function μ^c defined by: $\forall A \subseteq \Omega$, $\mu^c(A) = \mu(\Omega) - \mu(A^c)$.
- A set function μ is said to be concave or supermodular iff:

$$\forall A, B \subseteq \Omega,\ \mu(A \cup B) + \mu(A \cap B) \geq \mu(A) + \mu(B).$$

- A set function μ is said to be additive iff:

$$\forall A, B \subseteq \Omega,\ \mu(A \cup B) + \mu(A \cap B) = \mu(A) + \mu(B).$$

3 Operator based aggregation

3.1 Operators

An operator is a set function $\mu_{\boldsymbol{\varphi}}$ of Ω entirely defined by a vector $\boldsymbol{\varphi} \in \mathbb{R}^N$– hereafter called *the kernel* of the operator – having the same dimension as Ω. We define two operators here: the linear operator and the MacSum operator.

Let $\boldsymbol{\varphi} \in \mathbb{R}^N$ be a vector.

• The **linear operator** $\lambda_{\boldsymbol{\varphi}}$ is defined by:

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

Obviously, the linear operator is additive, so its complementary operator is itself.

• The **MacSum operator** $\nu_{\boldsymbol{\varphi}}$ and its complementary operator $\nu^c_{\boldsymbol{\varphi}}$, introduced in [7], and defined as $\forall A \subseteq \Omega$:

$$\nu_{\boldsymbol{\varphi}}(A) = \max_{i \in A} \varphi_i^+ + \min_{i \in \Omega} \varphi_i^- - \min_{i \in A^c} \varphi_i^-, \tag{1}$$

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

As shown in [7], the MacSum operator is a concave set function.

There is a very interesting link between linear and MacSum operators. Let $\boldsymbol{\varphi}, \boldsymbol{\psi} \in \mathbb{R}^N$ be two vectors 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 MacSum-core (or simply the core) of a kernel $\boldsymbol{\varphi}$ as the subset $\mathcal{M}(\boldsymbol{\varphi}) \in \mathbb{R}^N$ of the kernels of Ω that are dominated by $\boldsymbol{\varphi}$:

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

3.2 Aggregation

Let $\boldsymbol{\varphi} \in \mathbb{R}^N$ be a vector of Ω used as a kernel.

Let $\mu_{\boldsymbol{\varphi}}$ a concave operator and $\boldsymbol{x} \in \mathbb{R}^N$ a real vector. Then we define $\mathcal{A}_\mu : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{IR}$ 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 $\boldsymbol{\varphi}$ 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 $\overline{y} = \overline{\mathcal{A}}_\mu(\boldsymbol{x}, \boldsymbol{\varphi}) = \check{\mathbb{C}}_{\mu_{\boldsymbol{\varphi}}}(\boldsymbol{x})$, $\check{\mathbb{C}}$ being the discrete asymmetric Choquet integral [1].

We thus define:

• The linear aggregation.

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.

• The MacSum aggregation.

$\mathcal{A}_\nu(\boldsymbol{x}, \boldsymbol{\varphi}) = [y] = [\underline{y}, \overline{y}] = [\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\varphi}}}(\boldsymbol{x}), \check{\mathbb{C}}_{\nu_{\boldsymbol{\varphi}}}(\boldsymbol{x})]$.

Given the link between the linear and MacSum operators, we have:

$$\forall \boldsymbol{\varphi} \in \mathbb{R}^N, \forall \boldsymbol{x} \in \mathbb{R}^N, \forall \boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi}), \mathcal{A}_\lambda(\boldsymbol{x}, \boldsymbol{\psi}) \in \mathcal{A}_\nu(\boldsymbol{x}, \boldsymbol{\varphi}). \tag{3}$$

The values of $\underline{y}$ and $\overline{y}$ can be obtained by [2]:

$$\overline{y} = \sum_{k=1}^{N} \varphi^+_{\lfloor k \rfloor} \cdot \left(\max_{i=1}^{k} x_{\lfloor i \rfloor} - \max_{i=1}^{k-1} x_{\lfloor i \rfloor} \right) + \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} \cdot \left(\min_{i=1}^{k} x_{\lceil i \rceil} - \min_{i=1}^{k-1} x_{\lceil i \rceil} \right), \tag{4}$$

$$\underline{y} = \sum_{k=1}^{N} \varphi^+_{\lfloor k \rfloor} \cdot \left(\min_{i=1}^{k} x_{\lfloor i \rfloor} - \min_{i=1}^{k-1} x_{\lfloor i \rfloor} \right) + \sum_{k=1}^{N} \varphi^-_{\lceil k \rceil} \cdot \left(\max_{i=1}^{k} x_{\lceil i \rceil} - \max_{i=1}^{k-1} x_{\lceil i \rceil} \right), \tag{5}$$

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

Equations (5) and (4) are easy to derive w.r.t. the kernel (see [2]):

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

$$\frac{\delta \overline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k} = \left(\max_{i=1}^{l} x_{\lfloor i \rfloor} - \max_{i=1}^{l-1} x_{\lfloor i \rfloor} \right) + \left(\min_{i=1}^{u} x_{\lceil i \rceil} - \min_{i=1}^{u-1} x_{\lceil i \rceil} \right), \tag{6}$$

$$\frac{\delta \underline{\mathcal{A}}_\nu(\boldsymbol{x}, \boldsymbol{\varphi})}{\delta \varphi_k} = \left(\min_{i=1}^{l} x_{\lfloor i \rfloor} - \min_{i=1}^{l-1} x_{\lfloor i \rfloor} \right) + \left(\max_{i=1}^{u} x_{\lceil i \rceil} - \max_{i=1}^{u-1} x_{\lceil i \rceil} \right). \tag{7}$$

3.3 Extending operator-based aggregation to interval data

Extending linear aggregation to intervals is fairly straightforward. Let $\overline{\underline{\boldsymbol{x}}} \in \mathbb{IR}^N$ be an interval-valued vector of Ω and $\boldsymbol{\varphi} \in \mathbb{R}^N$ be a vector of Ω used as a kernel, we can define:

$$\begin{aligned}\mathcal{A}_\lambda(\overline{\underline{\boldsymbol{x}}}, \boldsymbol{\varphi}) &= \{\mathcal{A}_\lambda(\boldsymbol{x}, \boldsymbol{\varphi}) \ / \ \boldsymbol{x} \in \overline{\underline{\boldsymbol{x}}}\} = \big[\inf_{\boldsymbol{x}\in\overline{\underline{\boldsymbol{x}}}} \mathcal{A}_{\lambda_\psi}(\boldsymbol{x}), \sup_{\boldsymbol{x}\in\overline{\underline{\boldsymbol{x}}}} \mathcal{A}_{\lambda_\psi}(\boldsymbol{x})\big], \\ &= \big[\mathcal{A}_\lambda(\boldsymbol{x}_*, \boldsymbol{\varphi}), \mathcal{A}_\lambda(\boldsymbol{x}^*, \boldsymbol{\varphi})\big].\end{aligned} \tag{8}$$

where $\boldsymbol{x}^*$ and $\boldsymbol{x}_*$ are the vectors of $\mathbb{R}^N$ such that $\forall i \in \Omega,\ \ x_i^* = \overline{x}_i,\ \ x_{*i} = \underline{x}_i$ if $\varphi_i \geq 0$ and $\ x_i^* = \underline{x}_i,\ \ x_{*i} = \overline{x}_i$ if $\varphi_i < 0$.

As presented in [3], extending MacSum aggregation to intervals is rather straightforward too. In fact, there are two possible ways of building this extension: the disjunctive aggregation and the conjunctive aggregation.

The disjunctive aggregation is conservative and tries not to reject any information. It can be set as:

$$\begin{aligned}\mathcal{D}_\nu(\overline{\underline{\boldsymbol{x}}}, \boldsymbol{\varphi}) &= \bigcup\nolimits_{\boldsymbol{x}\in\overline{\underline{\boldsymbol{x}}}} \mathcal{A}_\nu(\boldsymbol{x}, \boldsymbol{\varphi}) = \{\mathcal{A}_\lambda(\boldsymbol{x}, \boldsymbol{\psi}) \ / \ \boldsymbol{x} \in \overline{\underline{\boldsymbol{x}}},\ \boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi})\}, \\ &= \{\mathcal{A}_\lambda(\boldsymbol{x}, \boldsymbol{\psi}) \ / \ \boldsymbol{x} \in \overline{\underline{\boldsymbol{x}}}, \psi \in \mathcal{M}(\boldsymbol{\varphi})\}, = \big[\underline{\mathcal{A}}_\nu(\boldsymbol{x}_*, \boldsymbol{\varphi}), \overline{\mathcal{A}}_\nu(\boldsymbol{x}^*, \boldsymbol{\varphi})\big].\end{aligned} \tag{9}$$

The conjunctive aggregation tries to reduce the set of values to those for which each set being aggregated agrees. It can be set either as:

$$\mathcal{C}_\nu^{\triangleleft}(\overline{\underline{\boldsymbol{x}}}, \boldsymbol{\varphi}) = \bigcap_{\boldsymbol{x}\in\overline{\underline{\boldsymbol{x}}}} \mathcal{A}_\nu(\boldsymbol{x}, \boldsymbol{\varphi}) = \bigcap_{\boldsymbol{x}\in\overline{\underline{\boldsymbol{x}}}} \{\mathcal{A}_\lambda(\boldsymbol{x}, \boldsymbol{\psi}) \ / \ \boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi})\}, \text{ or as:} \tag{10}$$

$$\mathcal{C}_\nu^{\triangleright}(\overline{\underline{\boldsymbol{x}}}, \boldsymbol{\varphi}) = \bigcap_{\boldsymbol{\psi}\in\mathcal{M}(\boldsymbol{\varphi})} \mathcal{A}_\lambda(\overline{\underline{\boldsymbol{x}}}, \boldsymbol{\psi}) = \bigcap_{\boldsymbol{\psi}\in\mathcal{M}(\boldsymbol{\varphi})} \{\mathcal{A}_\lambda(\boldsymbol{x}, \boldsymbol{\psi}) \ / \ \boldsymbol{x} \in \overline{\underline{\boldsymbol{x}}}\}. \tag{11}$$

Equation (10) means that the conjunction consists of intersecting all the intervals produced by the MacSum aggregation for each possible entry contained in the interval $\overline{\underline{\boldsymbol{x}}}$ while equation (11) means that the conjunction consists of intersecting all the intervals produced by linear aggregation for each $\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\varphi})$.

Both interpretations lead to:

$$\mathcal{C}_{\nu_{\boldsymbol{\varphi}}}(\overline{\underline{\boldsymbol{x}}}) = \Big[\min\big(\underline{\mathcal{A}}_\nu(\boldsymbol{x}^*, \boldsymbol{\varphi}), \overline{\mathcal{A}}_\nu(\boldsymbol{x}_*, \boldsymbol{\varphi})\big), \max\big(\underline{\mathcal{A}}_\nu(\boldsymbol{x}^*, \boldsymbol{\varphi}), \overline{\mathcal{A}}_\nu(\boldsymbol{x}_*, \boldsymbol{\varphi})\big)\Big]. \tag{12}$$

It is straightforward that computing the derivative w.r.t. the kernel of both conjunctive and disjunctive approaches can easily be achieved by considering Equations (6) and (7).

3.4 Learning an operator based aggregation

Learning an operator based aggregation means that, based on a dataset of M input-output pairs $\big\{(\boldsymbol{x}^j, y_j)\big\}_{j=1\ldots M}$, it may be possible to find a kernel $\hat{\boldsymbol{\varphi}} \in \mathbb{R}^N$ that ensures that the value $\mathcal{A}_\mu(\boldsymbol{x}^j, \hat{\boldsymbol{\varphi}})$ is as close as possible to $y_j\ \forall j \in \{1, \ldots, M\}$

(where μ can be either λ or ν). The most common method consists of minimizing, for the entire database, the quadratic difference between the prediction given by the aggregation function and the measurement. For the linear modelling, this can easily be achieved iteratively using the gradient descent method.

Regarding the MacSum modelling, in [2] it has been proposed to minimize the quadratic distance between y_j and the center of the interval $\mathcal{A}_\nu(\boldsymbol{x}^j, \hat{\boldsymbol{\varphi}})$. We propose to use the same method, with the difference that the derivatives, used in the gradient descent, are calculated considering the extreme values x^* and x_*, according to the intervallist extension chosen.

In this work, since at least one value of the interval-valued input is expected to reduce the discrepancy between predicted and measured values, the conjunctive extension seems the most appropriate.

4 Experiments

We propose to evaluate the ability of the MacSum operator to take into account input data with missing values in order to learn its kernel. As the vast majority of operations in image processing are based on convolution operations (which can be assimilated to linear aggregations), we propose to learn the kernel of a linear convolution on the basis of a set of examples. To avoid favoring the linear approach too much, we propose to model an infinite-response convolution with a finite-response model. To achieve this, we compute the horizontal gradient of a set of images with the Shen-Castan operator [6], which is an infinite impulse response filter, and model it by a convolution over a 5×5 neighborhood. In this experiment, we show that learning can still be performed even in the event of partial contamination of the database by missing data.

4.1 Data-set

As with article [3], we used a thousand 600×600 natural images sourced from the CLEF[3] project (see e.g. Figure (1)). The Shen-Castan horizontal component of the gradient has been computed using $a_0 = 0.3$ as a spread parameter. For each experiment, we randomly selected 60 images from the 1000 images in the database and randomly selected again 100 pixels, producing 6000 samples for each experiment.

For each sample, we considered the 5×5 neighborhood of the original image (for the input vector) and the corresponding value of the horizontal component of its gradient (for the output value). Each database element is therefore made up of an input vector of 25 integer values ranging in $[0, 255]$ and a signed real output value. Centered random Gaussian noise has been added to the output value, with a standard deviation of 30% of the standard deviation of the corresponding gradient image.

In this experiment, we proposed to learn the kernel associated with derivating the image within both linear and MacSum aggregation modeling.

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

Fig. 1. Four out of the 1000 images used for this experiment.

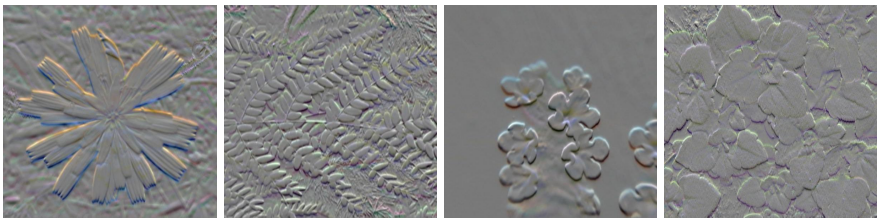

Fig. 2. The Shen-Castan derivative of the four images depicted in Figure (1).

We have divided the database into three parts of 2000 samples. The samples of the first third are assumed to be with no missing values. We call this set *the uncontaminated training data-set*. In the second third, certain values of the input data vector are assumed to be missing. The number of missing data items in each input vector is fixed for each experiment. On the other hand, the choice of which element of the input vector (among the 25) is missing has been randomly selected beforehand. We call this set *the contaminated training data-set*. We also call *the complete training data-set* the data-set obtained by supplementing the uncontaminated training data-set by the contaminated training data-set. The last third of the database has been used to test the quality of the learning. We call this set *the test data-set*.

4.2 How can missing values be accounted for?

In image processing, when a piece of data is missing or corrupted (e.g. in the case of impulse noise), it's common to use information from its neighborhood to assess the missing value. It is also possible to infer the missing value(s) by considering the complete data whose values are close to the known values of the contaminated vector (this is what is used in in-painting). In this case, however, the fact that the missing value is completely unknown is not expressed at all. As far as linear aggregation is concerned, the only way to represent the fact that the missing value is unknown is to give it an arbitrary value. We tested two methods, one consisting in giving a random value in the range $[0, 255]$, the other in systematically giving the same value, which in this case would be the center

of this range, i.e. 127.5. As we found no significant difference in the behavior of the estimate when choosing one or other of these methods, we opted for the simplest, i.e. to systematically give the value 127.5 to missing values.

When it came to MacSum aggregation, we had two options. The first was to give a missing value an arbitrary value, as with linear aggregation. The second was to replace a missing value by the interval $[0, 255]$. We present these two solutions for comparison.

4.3 Running the experiment

The aim was to determine whether the information provided by the contaminated data-set can be used to improve learning in the same way as a data complement without missing values.

Each experiment consisted in generating a database of 6000 samples and dividing it into three subsets as explained in section 4.1. For each experiment, we arbitrarily performed 200 iterations of the learning algorithm for both models (additive and MacSum), having found that each algorithm converged well for this number of iterations. For each model, we carried out the training with, firstly, the uncontaminated training data-set, then the complete training data-set. We observed the improvement, or deterioration, of learning by calculating the Pearson coefficient of determination R^2 using the test data-set. This experiment was carried out 100 times for four different levels of contamination (namely $1/25, 4/25, 8/25$ and $12/25$). This experiment has been run 100 times.

4.4 Results

To make reading the results easier, we propose two types of visualization.

In Figure (3) we present four illustrations where each point has as its abscissa the R^2 value obtained using only the uncontaminated training data-set and as its ordinate the R^2 value obtained with the complete training data set. Each Figure corresponds to a different level of contamination. Results obtained by learning the MacSum modeling are plotted in red, and those obtained by learning the linear modeling are plotted in blue. Results obtained by representing a missing value by an interval are plotted with a circle $\circ$ and those obtained by representing missing values by arbitrary values are plotted with a star $*$. We have also drawn the unit slope line in green.

The reading is the following. Any point above the line (in green) is symptomatic of increased learning by supplementing the uncontaminated base with the contaminated base. Any point below the unit line is symptomatic of a deterioration in learning by supplementing the uncontaminated base with a contaminated base.

There are several facts to be noted when looking at these Figures.

– First, using only uncontaminated data, the linear model is better able than the MacSum model to represent a linear system.

– Second, supplementing uncontaminated data with contaminated data to learn a linear model always results in degraded learning. This is also true for a

MacSum model if the missing value are replaced by an arbitrary value. On the other hand, if missing values are replaced by their interval of variation, learning performance improves, even with a contamination rate approaching 50%.

– Third, as contamination increases, the rate of increase in learning performance due to the use of contaminated data decreases.

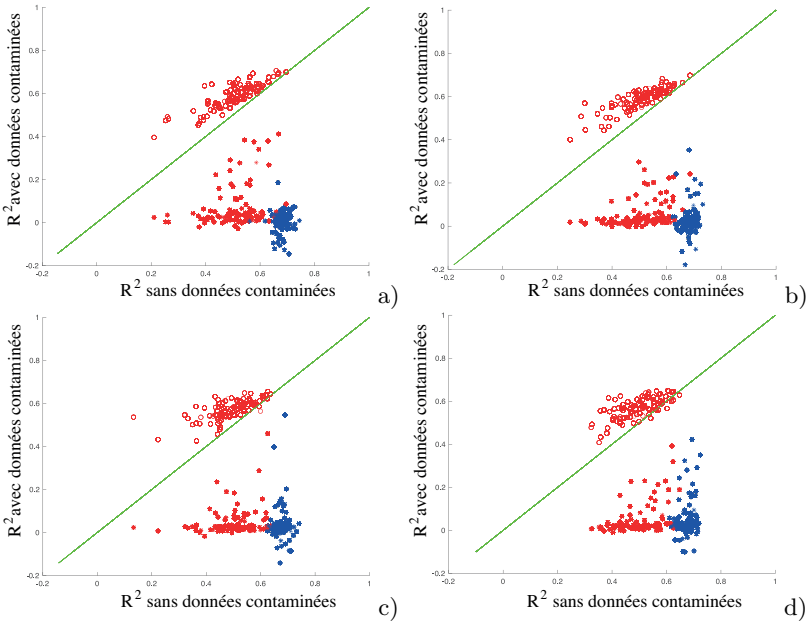

Fig. 3. Comparison of learning with and without missing values, with missing values rate of 4% (a), 16% (b), 32% (c), 48% (d)

Contamination rate	4%	16%	32%	48%
Linear without contaminated data	0.68	0.68	0.68	0.68
Linear with contaminated data	< 0.01	0.02	0.04	0.05
MacSum without contaminated data	0.51	0.51	0.49	0.50
MacSum without contaminated data	0.59	0.59	0.57	0.57

Table 1. Mean value of Pearson coefficient R^2 for different rates of contamination.

Table 1 gives the mean R^2 values for each experiment, while Table 2 shows the number of times the use of contaminated data improved learning. For MacSum modeling, we report in these tables only the approach of modeling the missing value by an interval. As can be clearly seen, the approach of replacing missing

Contamination rate	4%	8%	32%	48%
MacSum	100%	100%	96%	93%
Linear	0%	0%	0%	0%

Table 2. Percentage of experience where the R^2 coefficient has been improved.

values with arbitrary values does not improve learning capabilities (either for the linear approach or for the MacSum approach). On the other hand, replacing missing values with their range of possible values enables the MacSum approach to take advantage of the information available in the auxiliary data. Naturally, this ability diminishes somewhat as the level of contamination increases.

Table 1 gives the mean R^2 values for each experiment while Table 2 gives the number of times the use of contaminated data improved learning. For the MacSum modeling, we report in these tables only the approach of modeling the missing value by an interval. As can be clearly observed, the approach of replacing missing values with arbitrary values does not improve learning capabilities (either for the linear- or for the MacSum approach) as all $\star$ points are well below the green diagonal. On the other hand, replacing missing values with their range of possible values allows the MacSum approach to take advantage of the information available in the auxiliary data. Obviously, this ability diminishes somewhat as the level of contamination increases.

5 Conclusion

Learning a parametric model from an input-output database generally involves estimating a parameter to minimize a measure of compatibility between the output predicted by the model and the corresponding output of the database. In this context, when certain input vector values are missing, it is generally preferable to remove the contaminated data from the database. In this article, we propose to take advantage of the intervallist nature of the MacSum operator to include data with miss ing values in the training database. We have shown on an example that this choice was appropriate, as the addition of such data to the learning base improves its performance (in the sense of the linear coefficient of determination). However, this article raises more questions than it answers. For example, the choice of the minimized criterion for learning is perhaps a little simplistic, and it would be interesting to develop a learning method more in line with the intervallist nature of both the operator and the data.

Acknowledgment

The authors would like to thank Dorian Kauffmann for his useful remarks and comments.

References

1. Grabisch, M., Sugeno, M., Murofushi, T.: Fuzzy measures and integrals: theory and applications. Heidelberg: Physica (2000)
2. Hmidy, Y., Rico, A., Strauss, O.: Macsum aggregation learning. Fuzzy Sets and Systems **459**, 182–200 (2023)
3. Hmidy, Y., Rico, A., Strauss, O.: Extending the Macsum Aggregation to Interval-Valued Inputs. In: SUM 2022 - 15th International Conference Scalable Uncertainty Management. Lecture Notes in Computer Science, vol. 13562, pp. 338–347. Springer International Publishing, Paris, France (2022)
4. Loquin, K., Strauss, O.: Histogram density estimators based upon a fuzzy partition. Statistics and Probability Letters **78**(13), 1863–1868 (September 2008)
5. Loquin, K., Strauss, O.: On the granularity of summative kernels. Fuzzy Sets and Systems **159**(15), 1952–1972 (August 2008)
6. Shen, J., Castan, S.: Towards the unification of band-limited derivative operators for edge detection. Signal Processing **31**(2), 103–119 (1993)
7. Strauss, O., Rico, A., Hmidy, Y.: Macsum: a new interval-valued linear operator. International journal of approximate reasoning **145**, 121–138 (2022)