International Journal of Computational Intelligence Systems (2022) 15:89
https://doi.org/10.1007/s44196-022-00146-w

RESEARCH ARTICLE

# Measure Identification for the Choquet Integral: A Python Module

**Ezgi Türkarslan**[1,2] · **Vicenç Torra**[3]

Received: 20 July 2022 / Accepted: 28 September 2022


**Abstract**
Fuzzy integrals are common concepts which are used to aggregate input values in practical applications. Aggregation of inputs using fuzzy integrals opens up numerous possibilities for modeling interaction, redundancy, and synergy of inputs. However, fuzzy integrals need a fuzzy measure to start this aggregation process. This situation pushes us into the fuzzy measure identification process. This process becomes difficult due to the monotony condition of the fuzzy measure and the exponential increase on the number of measure parameters. There are in the literature many ways to determine fuzzy measures. One of them is learning from data. In this paper, our aim is to introduce a new fuzzy measure identification tool to learn measures from empirical data. It is a Python module which finds the measure that minimizes the difference between the computed and expected outputs of the Choquet integral. In addition, we study some properties of the learning process. In particular, we consider $k$-additive fuzzy measures and belief functions as well as arbitrary fuzzy measures. Using these variety of measures we examine the effect of $k$ and noisy data on the learning process.

**Keywords** Fuzzy measure identification · $k$-additive fuzzy measure · Belief functions · Möbius transform · Python

## 1 Introduction

Fuzzy integrals [1–3] are used in applications as a useful tool for data aggregation. Their advantage in aggregation is that they permit to model interactions between information sources by means of fuzzy measures. This is not the case when using e.g. arithmetic means, weighted means, and all other types of generalized means. Note that in means we can express our background knowledge on the information sources by means of weights. Nevertheless, these weights are assuming independence between the sources. In contrast, a fuzzy measure permits to express redundancy and complementarity among the sources (see e.g. [4]).

One of the most challenging problems [5, 6] in order to apply fuzzy integrals is the determination of fuzzy measures. They are set functions, so, this means, that given $n$ inputs we need about $2^n$ parameters. More particularly, we need $2^n - 2$ parameters as we have boundary conditions for the empty set and the full set. Since about 2000 there have been research on how to identify or learn these measures from data or elicit them from experts.

Some research focuses on supervised techniques. That is, measure identification problems [7–10] in which one has access to pairs of input–output data. In other words, the problem is to consider a data set consisting of a set of inputs and their expected output, the measure is identified so that the output is approximated given the input and a particular fuzzy integral. The input–output pairings may be observed from experiments or may be simply given by subject matter experts. Solutions differ depending on which fuzzy integral is used in the model, which error function or objective function is considered in the problem, and which additional constraints are added into the problem. For example, when the integral is the Choquet integral, the problem can be formalized as a quadratic optimization problem with linear constraints. An example of additional constraint to the problem

Vicenç Torra have contributed equally to this work.

✉ Ezgi Türkarslan
ezgi.turkarslan@tedu.edu.tr

Vicenç Torra
vtorra@cs.umu.se

[1] Department of Mathematics, Ankara University, Faculty of Science, Dögol road, Ankara 06560, Ankara, Turkey

[2] Department of Mathematics, TED University, Faculty of Arts and Science, Ziya Gökalp road, Ankara 06420, Ankara, Turkey

[3] Department of Computing Sciences, Umeå University, MIT building, Umeå 901 87, Västerbotten, Sweden

Published online: 21 October 2022

is considering entropy or minimum variance of the fuzzy measure as in [11]. Solution of the problem using genetic algorithms is given in [12]. A review of methods is provided in [13] and [14]. Software for solving the problem is available in KappaLab [15] (last version from 2015), a package for R which provides several functions to operate with fuzzy measures and integrals. For example, the package provides fuzzy integrals (including Sugeno and Choquet integrals), identification of measures for the Choquet integral (including least squares error and variations as the minimum variance), the Shapley value and Möbius transform.

We have developed an alternative module for Python, which is one of the most used programming language nowadays. The module provides Sugeno and Choquet integrals, functions for computing Shapley values and interaction indices, transforms (Möbius and (max, +)-transform), as well as measure identification for the Choquet integral. In this paper, we present the software focusing on the identification of measures for the Choquet integral. We can learn an arbitrary fuzzy measure, but also restrict the measure to be a belief function, or to be a $k$-additive measure for a given $k$. Both restrictions are also possible (i.e., a $k$-additive measure for a given $k$ which is also a belief measure).

The identification of a fuzzy measure needs to deal with the problem of noisy data. We also study this problem using our new software. In particular, we study the effect of noise in the identification of the fuzzy measure, and we also study how a selection of a constrained fuzzy measure (i.e., $k$-additive or belief) affects the effectiveness of the output). In other words, we study how different $k$ leads to different measures and how suitable they are for approximating the original data. We discuss the selection of the right complexity for fuzzy measure selection. In other words, this is about the selection of a correct $k$ for a $k$-additive fuzzy measure.

The structure of the paper is as follows. In Section 2, we present some preliminaries. In Section 3, we discuss the problem of fuzzy measure identification. We describe the standard formulation in terms of an optimization problem and discuss how we build the equations in our case. In Section 4, we describe our experiments and results. The paper finishes with some conclusions and research directions.

## 2 Preliminaries

In this section, we will briefly describe a few key concepts related to fuzzy measure and optimization, which will be directly applied in the following sections.

**Definition 1** [1] Let $\Omega$ be a finite set and let $P(\Omega)$ be power set of $\Omega$. If

i .)$\mu(\varnothing) = 0$,

ii .)$\mu(\Omega) = 1$,

iii .)$A \subseteq B$ implies $\mu(A) \leq \mu(B)$ (monotonicity),

then the set function $\mu : P(\Omega) \rightarrow [0, 1]$ is called a fuzzy measure on $\Omega$ .

The Möbius representation of a fuzzy measure is important to model interactions between criteria.

**Definition 2** [16] The Möbius representation (Möbius transform) of a set function $\mu$ on $\Omega$ is a set function $m : P(\Omega) \rightarrow \mathbb{R}$ defined by

$$m(A) := \sum_{B \subset A} (-1)^{|A \setminus B|} \mu(B). \tag{1}$$

Given a Möbius function $m$, we can build the fuzzy measure $\mu$ using the following equation:

$$\mu(A) = \sum_{B \subset A} m(B) \tag{2}$$

for all $A \in P(\Omega)$.

It is easy to see that the Möbius representation over singletons is equal to the fuzzy measure itself. It can also be proven that given $\mu$, if we compute the Möbius transform using Equation (1), the measure that results from Equation (2) is precisely $\mu$. There are other transforms in the literature with similar properties. For example, the (max, +) is one of them.

Fuzzy measure identification is quite computationally costly since it is defined on the power set. To overcome this problem, there have been several attempts to define families of measures with reduced complexity. For example, Grabisch [17] proposed the concept of $k$-additive fuzzy measure. We define it below because we are using this family of measures in the paper. This family has the property that varying the parameter $k$, we range from a probability (when $k = 1$) to an arbitrary fuzzy measure (when $k = |\Omega|$). A previous definition of a family of measures with reduced complexity is the one defined by Sugeno (known as Sugeno $\lambda$-measures). A related approach to $k$-order additive are its generalizations via t-conorms (see e.g. [18]).

**Definition 3** [19] Let $\Omega$ be a finite set and let $\mu$ be a fuzzy measure on $\Omega$. $\mu$ is said to be $k$-additive if its Möbius transform $m$ satisfies $m(A) = 0$ for all $A \subset \Omega$ such that $| A |> k$ and there exist at least one subset $A \subset \Omega$ with $| A |= k$ such that $m(A) \neq 0$.

For $k$-additive measures, the value of $k$ corresponds to its complexity. When $k = 1$ we have that the measure is additive and, thus, the Möbius transform on the singletons define a probability distribution. No interactions are present. Then, when we increase $k$ and set it to $k = 2$, we have interactions between pairs of elements in the reference set $\Omega$. The larger

the $k$, the larger the interactions. Naturally, this means that we have an increasing number of parameters. Note that for $k = 1$ we have only $|\Omega|$ parameters to identify (i.e., the values on the singletons). For $k = 2$ we have $|\Omega|$ parameters corresponding to the singletons plus $(|\Omega| \times (|\Omega| - 1))/2$ parameters corresponding to the interactions between pairs of objects. Then, for $k = |\Omega|$ we naturally have $2^{|\Omega|} - 2$ parameters.

Not all arbitrary set functions are Möbius transforms of a measure. The following theorem characterizes when this holds.

**Theorem 1** [16] *Let $\Omega \neq \emptyset$ be a finite set and let $\mu : P(\Omega) \to \mathbb{R}$ be a set function. Then, $\mu$ is a fuzzy measure on $\Omega$ if and only if its Möbius representation m satisfies*

$$i\ m(\emptyset) = 0,$$

$$ii \sum_{B \subset \Omega} m(B) = 1,$$

$$iii \sum_{x \in B \subset A} m(B) \geq 0, \text{ for all } A \subset \Omega \text{ and for all } x \in A.$$

To integrate a function with respect to a fuzzy measure we can use fuzzy integrals. The Choquet integral is one of them, another one is the Sugeno integral. The concept of the Choquet integral can be considered as a generalization of the weighted average where, conceptually, the main difference is that now we assign a weight to each subset of the universal set by means of the fuzzy measure instead of only assigning weights to the singletons as in the weighted average.

**Definition 4** [20] Let $\Omega = \{o_1, ..., o_n\}$ be a finite set and let $\mu$ be a fuzzy measure on $\Omega$. The Choquet integral of a function $f : \Omega \to [0, 1]$ with respect to $\mu$ is defined by

$$(C) \int_{\Omega} f \, d\mu := \sum_{k=1}^{n} \left( f(o_{(k)}) - f(o_{(k-1)}) \right) \mu(E_{(k)}), \tag{3}$$

where the sequence $\{o_{(k)}\}_{k=0}^{n}$ is the permutation of the sequence $\{o_k\}_{k=0}^{n}$ such that $0 = f(o_{(0)}) \leq f(o_{(1)}) \leq f(o_{(2)}) \leq ... \leq f(o_{(n)})$ and $E_{(k)} := \{o_{(k)}, o_{(k+1)}, ..., o_{(n)}\}$.

In applications, the problem of building data-driven models (i.e., learning models from data) is a crucial one. The identification of fuzzy measures from a training data set has always been a difficult problem for practical use of fuzzy measures. It is well known that minimizing a squared error criterion results in a quadratic program. We will describe this problem.

Let $X$ be the set of examples to be used in the learning process, corresponding to $m$ examples, records or instances. Let $x^j$ represent the *jth* example. Each example is described in terms of $n$ attributes, variables or criteria. Let $Y = (y_1, ..., y_m)$ be the expected output for these examples. We use $P$ to denote the decision matrix that includes both inputs $X$ and outputs $Y$: $P = [x_1^j ..., x_n^j \mid y^j]$ for each $j = 1, ..., m$. In this paper we consider the problem of fuzzy measure identification from the matrix $P$. It is a supervised approach, using the machine learning jargon.

More concretely, we use the Choquet integral as our model. Then, the integral is used to aggregate each *j*th example in the $P$ matrix. An error term is formed between the aggregated value and the expected outcome. By the minimizing error term, the fuzzy measure is identified.

Let $e^j = y^j - CI_\mu(x_1^j, ..., x_n^j)$ be the error term $y^j$ and between the Choquet integral with respect to fuzzy measure $\mu$. The identification of fuzzy measures by minimizing a quadratic error term is defined below. In addition to the error, we also need to consider some constraints related to the fuzzy measure. I.e., the set function we are searching needs to satisfy the constrains of a fuzzy measure.

$$\begin{aligned} &\min_{\mu} \sum_{j=1}^{m} (y^j - CI_\mu(x_1^j, ..., x_n^j))^2 \\ &s.t. \quad \mu(\emptyset) = 0, \\ &\qquad \mu(\Omega) = 1, \\ &\text{If } A \subseteq B \text{ then, } \mu(A) \subseteq \mu(B). \end{aligned} \tag{4}$$

This problem can be formulated as a quadratic optimization problem. The constraints related to the fuzzy measure are linear ones. Thus, the optimization problem can also be seen as follows for appropriate matrix $Q$, appropriate vector $q$ and appropriate matrix $R$ (see e.g. [21]).

$$\begin{aligned} &\min_{\mu} \frac{1}{2} \mu^t Q \mu + q^T \mu \\ &s.t. \quad \mu(\emptyset) = 0, \\ &\qquad \mu(\Omega) = 1 \\ &\qquad 0 \leq \mu \leq 1, \\ &\qquad R\mu \geq 0. \end{aligned} \tag{5}$$

## 3 Fuzzy Measure Identification

In this section, we describe how we solve this problem in practice in our software in Python. We will represent subsets of $\Omega$ in terms of their dyadic representation. That is, for $\Omega = \{o_1, o_2, o_3, o_4\}$ we represent subsets as (0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1), .... The first one represents the empty set, then (0, 0, 0, 1) represents $\{o_1\}$ as the last position can be understand as the bit for 1, (0, 0, 1, 0) corresponds to $\{o_2\}$, (0, 0, 1, 1) corresponds to $\{o_1, o_2\}$ and so on.

### 3.1 Implementation of the Solution

We follow here the approach previously described by Imai et al. [21]. See also e.g. [14, 22]. That is, we formulate the problem as a quadratic optimization problem with linear constraints as in Equation 5. The Choquet integral of a function with respect to a measure can be expressed as a vector multiplication of the Möbius transform of the measure and a vector built from the input. This formulation is advantageous for the identification problem. This is so because then the Choquet integral is just a linear combination. As a consequence, the objective function becomes a quadratic expression in terms of matrix multiplication. The optimization problem is then about finding the Möbius transform of the measure instead of finding the measure itself. We detail these steps in the next sections. First the objective function, and then we discuss the constraints.

### 3.2 Implementing the Choquet Integral

More formally, let $x^i$ be an input vector with $n$ values and let the function $f(o_j) = x^i_j$ for $\Omega = \{o_1, o_2, ..., o_n\}$, then the Choquet integral of $x^i$ can be expressed in terms of a $2^n - 1$ vector that we call $a^+_i$. Then, $a^+(A)$ for any $\emptyset \neq A \subset \Omega$ represents the value:

$$a^+(A) = \min_{o \in \Omega} f(o).$$

For example, given $f(o_1) = 11$, $f(o_2) = 3$, $f(o_3) = 7$, $f(o_4) = 11$, we have the vector $a^+_i = (11, 3, 3, 7, 7, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1)$. The linear product of $a^+$ and the Möbius transform leads to the Choquet integral. Here $a^+$ and the Möbius transform $m$ need to be correctly aligned. To make this alignment, in the software we use the correspondence above and the first position of $m^+$ corresponds to the set represented by (0, 0, 0, 1), second position to (0, 0, 1, 0), the third to (0, 0, 1, 1). We will use $m^+$ to represent the Möbius transform of all subsets of $\Omega$ except the empty set. More formally, given data $x^i$, if $a^+_i$ is the vector associated to data $x^i$, we can compute the Choquet integral of $x^i$ with respect to the Möbius transform $m^+$ of $\mu$ as follows: $CI_\mu(x^i) = a^+_i m^+$. That is, the Choquet integral is the product of two vector. This transformation is achieved in our program in python by `coefCIMobius` (which returns $a$ instead of $a+$ and thus including a zero associated to the empty set). The function receives $f(o_n), \ldots, f(o_1)$. So, in the order of the vector of bits.

```
>>> coefCIMobius([1,7,3,11])
[0, 11, 3, 3, 7, 7, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1]
```

### 3.3 Implementing the Objective Function

Let us consider again the error of the *ith* example:

$$e^i = y_i - CI_\mu(x^i) = y_i - a^+_i m^+.$$

We can express this expression in matrix form as $e = y - (a^+)^T m^+$. Then, the objective function, or estimation error can be expressed as follows:

$$\begin{aligned} MSE(m) &= \sum e_i * e_i = e^T e \\ &= (y - (a^+)^T m^+)^T (y - (a^+)^T m^+) \\ &= (y^T y - 2(a^+)^T y m^+ + m^+ (a^+)^T a^+ m^+). \end{aligned} \tag{6}$$

This expression is of the form $(1/2) z^T Q z + q^T z$. Note that $m^+$ plays the role of $z$, so $Q = 2(a^+)^T a^+$ and then $q^T = -2(a^+)^T y$.

We have two functions related to the objective function in our python module. One that given some data evaluates the objective function and returns its value. This function is called `ciModel_of`. The other function computes from the data the matrices $P$ and $q$, assuming that the variable is the vector of the Möbius transform $m^+$. This other function is called `buildOFMobius`.

### 3.4 Implementing the Inequality Constraints

The monotonicity condition establishes that for all $B \subset A$, we need to have $\mu(B) \leq \mu(A)$. This condition can be established equivalently for Möbius transform. In any case, from the point of view of the optimization problem, we do not need to consider all pairs of sets $A$, $B$ such that $B \subset A$. It is enough to consider sets $A$ and subsets $B$ that differ in only one element, say $x_0 \in \Omega$.

Then, it is also noticeable that given a set $B = A \setminus \{x_0\}$, we may assume that the measure of this set will be zero or positive. So, $\mu(A)$ will be larger than or equal to $\mu(B)$ when the Möbius transform of $\sum_{B' \subseteq B} \mu(B' \cup \{x_0\}) \geq 0$. That is, as an equation:

$$\begin{aligned} \mu(B) \leq & \mu(A) \\ &= \sum_{B' \subseteq A} m(B') = \sum_{B' \subseteq B} m(B') + \sum_{B' \subseteq A} m(B' \cup \{x_0\}) \\ &= \mu(B) + \sum_{B' \subseteq A} m(B' \cup \{x_0\}) \end{aligned} \tag{7}$$

This is the way we have used to define the monotonicity conditions in our implementation. Figure 1 represents these inequalities. We include $A$ and $B$ in the figure where

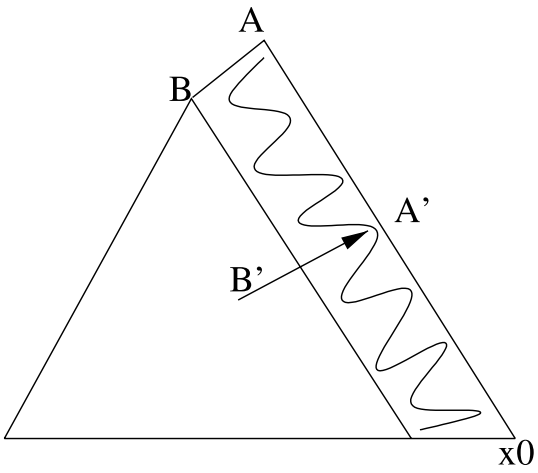


**Fig. 1** Graphical representation of the inequality constraints with respect to the Möbius transform. We consider all $B' \subseteq B$ and then $A' = B\{x_0\}$

$B = A \setminus \{x_0\}$, and also a subset $B'$ of $B$ which produces $A' = B' \cup \{x_0\}$ when adding the element $x_0$.

This approach naturally also works when $A$ is a singleton, therefore $B$ is the empty set, and, thus, $A = \{x_0\}$. Naturally, in this case, the inequality simply means that $m(\{x_0\}) \geq 0$.

Observe again that this construction works well because the property holds by induction. That is, the singletons will be positive, and then assuming that this works for sets $B$ then the inequality will work well for $A = B \cup \{x_0\}$.

As a summary, we include below the algorithm to define these inequalities. We can observe that in general for each set $A$ we need to consider all subsets $B$ of $A$. Thus, in general, the computational cost of this process is $\left(2^{|\Omega|}\right) \times \left(2^{|\Omega|}\right)$ which is $(2^{2|\Omega|})$.

Step 1 For all $A$,

Step 2 For all $x_0$ in $A$

Step 3 Add the following equation:

$$\sum_{A' \subseteq A \setminus \{x_0\}} m(A' \cup \{x_0\}) \geq 0 \tag{8}$$

This process is provided by the function `buildEquationsMobius (n)` where $n$ represents the number of elements in $\Omega$. This function returns the matrix with the left hand side of Equation 8 and a vector with the zeros in the right hand side of the same equation.

### 3.5 Variations: Belief Functions, k-Additive Measures, and Shapley Values

When we define the optimization problem it is easy to require that the measure is a belief measure, that it is $k$-additive, and that its Shapley value is of a given form. We have implemented these variations in our software. This is implemented as follows.

- To require that the solution is a belief function we need to add the following constraints:

  $m(A) \geq 0$ for all $A \subseteq \Omega$.

  This is provided by the function `buildEquationsMobiusPositive(n)`, that creates all required constraints given the number of elements $n$ in $\Omega$ (as for the other inequality constraints, the function returns the matrix corresponding to the left hand side of the equation and the vector corresponding to the right hand side of the equation).
- To require that the solution is a $k$-additive measure, we need to add the following constraints:

  $m(A) = 0$ for all $A$ such that $|A| \geq k$.

  In this case, it is the function `buildEquationsKAdditive(n, k)` which creates the corresponding matrix and vector. Naturally, we need the number of elements $n$ in $\Omega$ and also the value $k$.
- To require that the solution has a given Shapley value, we consider the expression of Shapley value given the Möbius transform. That is, the Shapley value corresponding to $x_i$ is $\sum_{x_i \in S} m(S)/|S|$. The function `buildEquationsShapley (n, shapley)` constructs the corresponding equations for a given vector of Shapley values (parameter `shapley`).

### 3.6 Implementation of Python

Our solution in python has been implemented by the function `ciSolveMSE` which given the data (input and expected output) returns the solution of the quadratic problem with linear constraints. We use the functions mentioned above to build the matrices of the objective function and the ones for the constraints. Then, we solve the problem using `cvxopt` and the function `solvers.qp`. This function returns the Möbius transform of the fuzzy measure, and then using the function `fromMoebius2FM` we can build the fuzzy measure itself. This corresponds to Definition 2.

The function that identifies a measure includes parameters that permit us to choose a belief function, a $k$-additive measure (or both), or just an arbitrary measure. By default, we identify an arbitrary measure. As an example, we can call this function with:

```
ciFindMoebius (data85, belief=True, kAdditive=3)
```

to solve an example called `data85` supplied in the software, which is taken from [22].

The software is available at [23].

## 4 Experiments

We have studied two different aspects related to the identification of fuzzy measures. First, we study the effect of $k$ for $k$-additive measures in terms of the error of test and training data. Second, we study how noise affects the learning of the measure identification problem. Third, we have applied our software to larger data sets to check the computation times.

### 4.1 Methodology

These studies are done using the following process.

- $\mu$ = Generate at random a fuzzy measure.
- $(X_{Tr}, X_{Te})$ = Generate a training and testing sets of given sizes ($nTr$, $nTe$). To generate these sets, we use uniform distribution on the space of data, which is the interval [0,10].
- $(y_{Tr}, y_{Te})$ = Find the outcome of both training and test data sets (using the Choquet integral with respect to $\mu$). That is, $y_{Tr} = CI_{\mu}(X_{Tr})$ and $y_{Te} = CI_{\mu}(X_{Te})$.
- $(y_{Tr}, y_{Te})$ = Add noise to both $y_{Tr}$ and $y_{Te}$. Noise follows a Gaussian distribution $N(0, nL)$ where $nL$ is the noise level. We have different noise levels for the test and the training sets. Say $nL_{Tr}$ and $nL_{Te}$.
- $\mu'$ = identify a fuzzy measure from the data $(X_{Tr}, y'_{Tr})$.
- $\hat{y}_{Te}$ = Estimate $y_{Te}$ using $\mu'$. That is, $\hat{y}_{Te} = CI_{\mu'}(X_{Te})$.
- $error$ = Compute the error between the $\hat{y}_{Te}$ and $y_{Te}$. That is,

$$error = ||y_{Te} - \hat{y}_{Te}||_2 = ||y_{Te} - CI_{\mu'}(X_{Te})||_2. \quad (9)$$

Due to the random factors, each execution of this procedure will lead to different results. Because of that, we repeat the whole process $nIt$ times and compute the average of the error.

### 4.2 The Effect of k

As we have stated above, for $k$-additive measures, the value of $k$ represents its complexity. From a machine learning perspective, the more parameters, the more we may have overfitting. It is crucial to have an appropriate selection of the complexity of the problem. That is, in the case of $k$-additive measures, to select an appropriate $k$. If $k$ is too small the complexity of the measure is not enough to represent the complexity of the data. If $k$ is too large, then we may be learning not only the data but also the error.

Because of that, we have studied the effect of $k$ in the learning process. In Figure 2 we represent the average error when we learn a measure from a set of 400 examples and 80

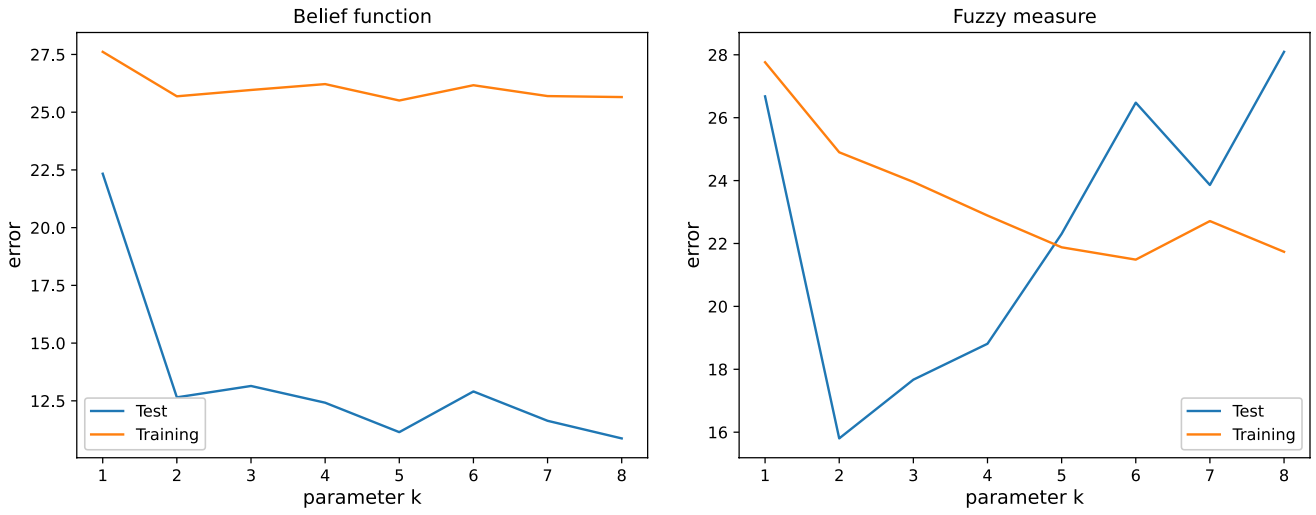


**Fig. 2** Error of the objective function (Equation 9) for the training and test sets when the measure is a belief function (left) and when it is an arbitrary measure (right). We use here Equation (9) to compute the error

examples are used as test. We used the approach described in Section 4.1 with 8 inputs, $nL_{Tr} = 2$ and $nL_{Te} = 0$ (i.e., zero error in the test data and proportional to two in the training data). Note that as the number of examples in the training set is 400 and 80 for the test, the error for test data should be about 5 times the one of train set.

- The figure shows a nice pattern when the measure learned is an arbitrary fuzzy measure. For the training data, the error decreases when we increase complexity. In contrast, for the test data error only decreases for $k = 1$ and $k = 2$. After this point, the test error increases. That is, we have that the optimal complexity is with $k = 2$ and larger $k$ does not help.
- For belief functions this pattern is not so clear. For the training set the error decreases when $k$ increases, but not so significantly. Then, for the test set, there is a clear improvement with $k = 2$ and from this complexity there is some improvement but not so significant. It seems somehow that $k = 2$ is also a good choice.

Therefore, for this data set it seems that a $k$-additive measure with $k = 2$ is the best alternative. This applies for both general fuzzy measures and belief functions. Larger values of $k$ can further reduce the error of the training set but it is not clear that the performance is better with respect to test sets. In addition, larger values of $k$ imply larger complexity in the definition of the fuzzy measure and its interpretability. Therefore, $k = 2$ provides a good trade-off between measure complexity and quality of the result.

### 4.3 The Effect of the Noise Level

We have also studied the effect of noise in measure identification. We consider values at $nL_{Tr} = 0.05, 1, 1.5, ..., 4$ and $nL_{Te} = 0$. Figure 3 shows the estimation error (difference

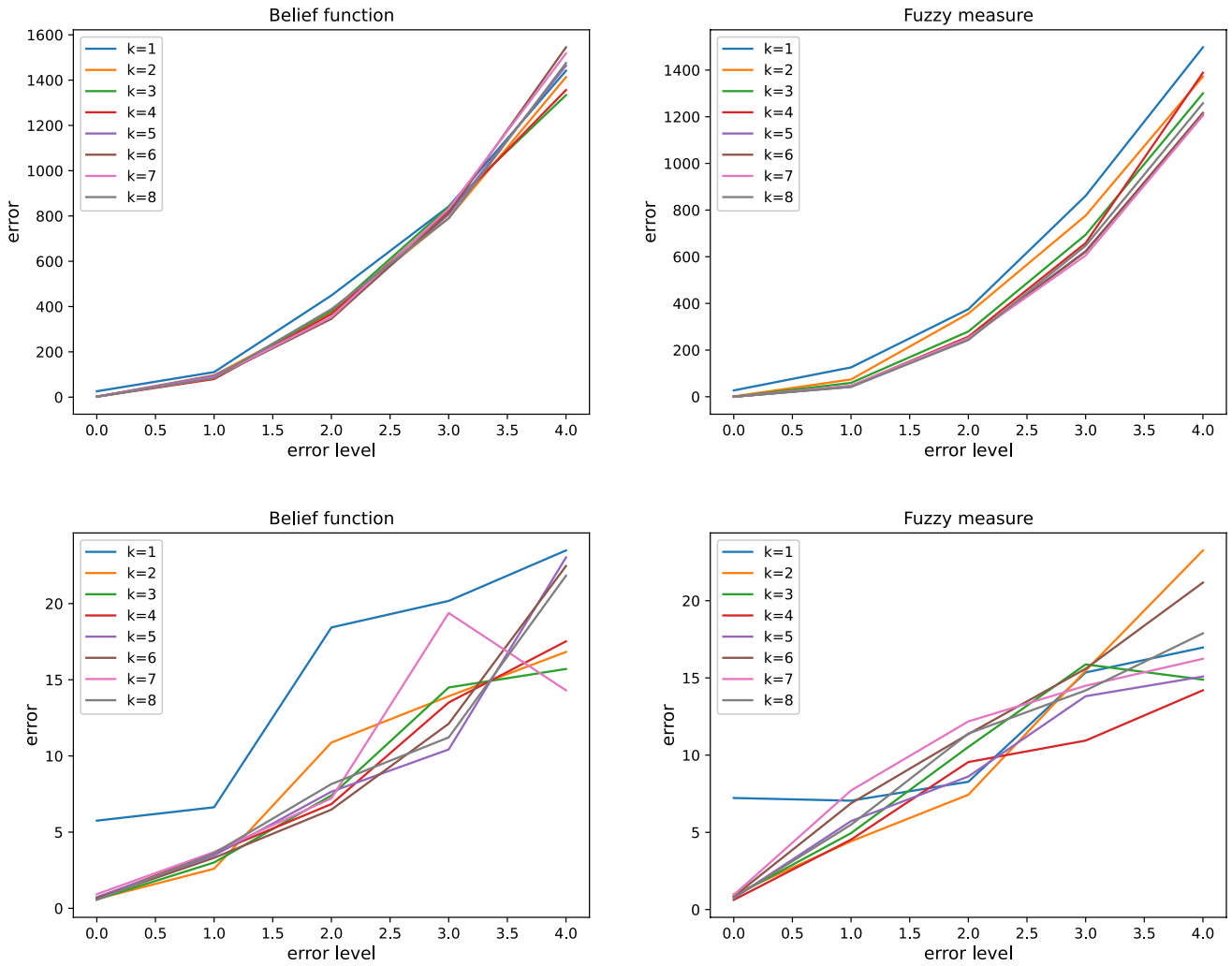


**Fig. 3** Error for the training (top) and test (bottom) sets when the measure is a belief function (left) and when it is an arbitrary measure (right). Mean of 10 executions

between expected and computed outcome – Equation 9) in terms of noise in the training data. we can see, as expected, that the estimated error steadily increases when the noise level increases. For the training data, this effect is mainly the same independently of the complexity of the measure complexity used ($k = 1, ..., k = 8$). In contrast, the results are not so similar in what respects test data.

For test data, recall that we use $nL_{Te} = 0$ – no noise. Then, we observe that there is a larger estimation error and that this estimation error is greater when we increase the noise level in the training set. That is, it is more and more difficult to identity the correct measure and the graphs does not seem to show a clear conclusion for which is the best complexity (value of $k$). For belief functions, $k = 4$ seems the one better suited as it has a more consistent low error.

Figure 4 displays the standard deviation of the error of the 10 same executions we have described above. We can see that, in general, when the noise level in the data increases the standard deviation of the error in the objective function also increases. It is also very clear that for $k = 1$ the standard deviation is rather large even with a noise level equal to zero. So, we may conclude here that $k = 2$ is the minimum value of $k$ that seems acceptable.

### 4.4 Belief Functions

The results displayed so far permit to make also a comparison of belief functions against arbitrary $k$-additive measures, and fuzzy measures in general. Belief functions seems to be less sensitive to overfitting. Figure 2 is the best one to visualize this fact. As we have discussed above, for an arbitrary fuzzy measure increasing $k$ reduces the training error but for values $k > 2$ the test error increases. The test error for the belief functions do not have the same behavior,

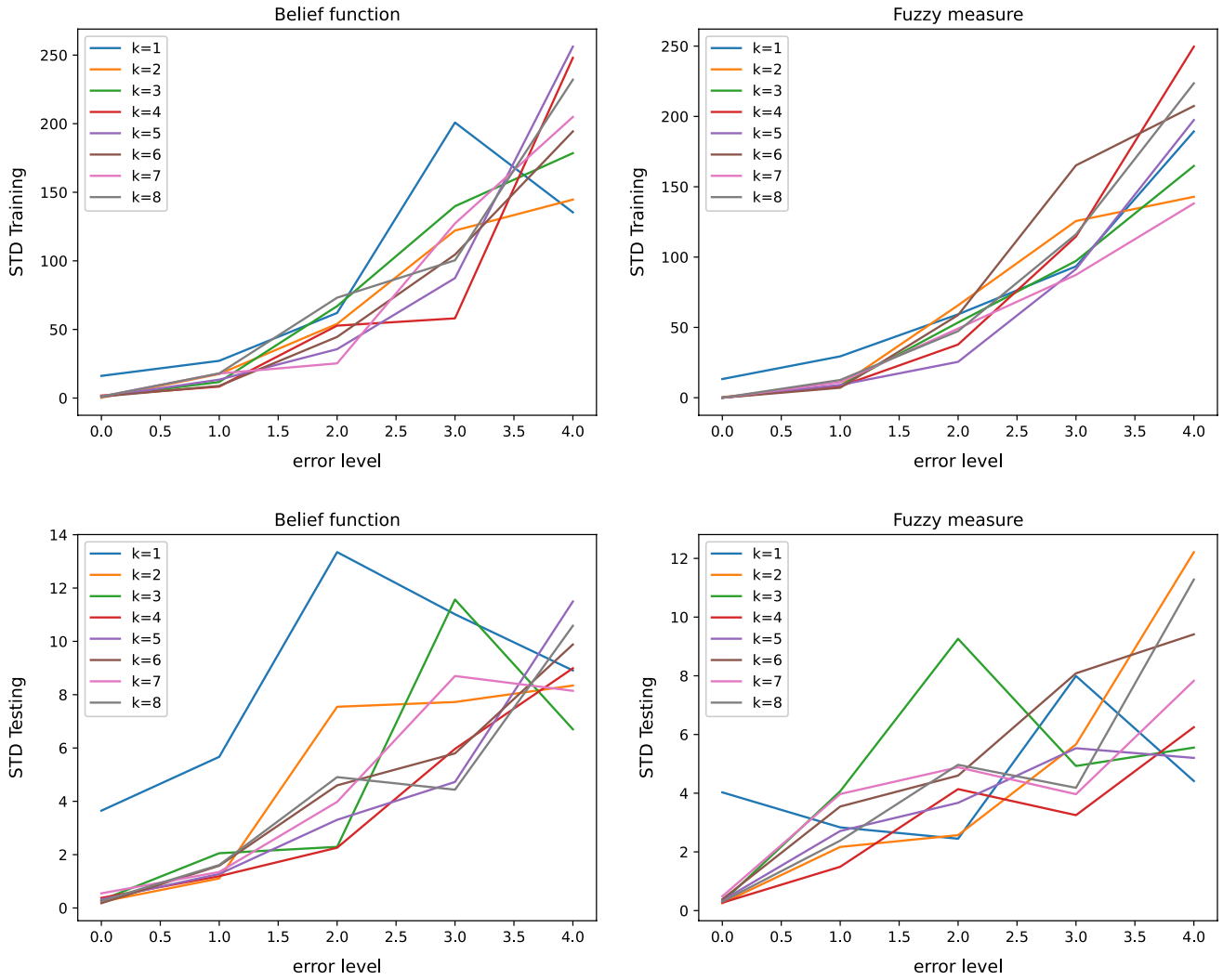


**Fig. 4** Standard deviations for the training (top) and test (bottom) sets when the measure is a belief function (left) and when it is an arbitrary measure (right). Mean of 10 executions

which is possibly because overfitting does not take place. This is naturally due to the fact that belief functions have less freedom than other measures as the Möbius should be always positive.

### 4.5 Larger Data Sets

We have applied our software to larger data sets available in `sklearn`. We have considered two regression problems, Diabetes and California. The first consists of 442 records and 10 inputs. The second one 20640 records and 8 inputs.

We have determined fuzzy measures for data sets with different number of records, from a small number of records to the full data set, and $k$-additive measures for different values of $k$. For example for the California data set, considering arbitrary fuzzy measures with either $k = 2$ or $k = 3$, we get in a regular laptop (Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz) execution times of less of 1 second for 10 records, between 1 and 5 seconds for 100 instances, and 30 seconds for 1000 instances, and 940 seconds for the full set of 20640 instances. For the California data set, learning the measure takes longer, as expected, as the number of parameters to identify goes from $2^8 - 2$ to $2^{10} - 2$. More precisely, it takes about 70 seconds for identifying a measure for the full data set of 442 instances.

## 5 Conclusions and Future Work

In this paper, we present a Python package that includes several alternatives for learning fuzzy measures from data. We also describe two types of experiments conducted using this package. The first one is for $k$-additive fuzzy measures: we know that as $k$ increases, the number of interactions increases between criteria. It means that as $k$ changes, the noise level in the numerical application changes. We investigate the effect of $k$ on the noise level. As a result, we obtain that the training error decreases when the complexity of the model increases. Nevertheless, with respect to the test data, the error decreases only for $k = 1$ and $k = 2$. The test error increases after this point. That is, we can see that $k = 2$ is the optimal complexity and that larger $k$ does not help. Among the measures studied, this is more clearly seen with belief functions. The second experiment is about the effects of noise in learning. We have seen, as expected, that the noise on the data makes more difficult the identification of the measure. We have also seen that the variance on the square error increases when the noise in the training data increases. The experiments also show that the standard deviation of the square error on different executions can also help on selecting a good parameter $k$ in $k$-additive measures.

The Python module provides some basic functions to operate with fuzzy measures. This includes Sugeno and Choquet integral, Möbius and (max, +)-transform, and measure identification for the Choquet integral. One of the limitations of the software is that it does not provide a solution for the Sugeno integral. This is a non-linear and non-quadratic problem. We leave as future work to provide functions for measure identification for the Sugeno integral, as well as considering additional constraints on the identified measure. For example, sparsity conditions for the fuzzy measure for models based on the Choquet integral.

Our package is available at [23].

**Author Contributions** Both authors contributed in the design of the research, the experiments, and the analysis of the results. Text was written and revised by both authors. Software was written by VT.

**Funding** This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Vicenç Torra is supported by WASP. Ezgi Türkarslan has been supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) 2214 Doctoral Research Grant 1059B142100223.

**Data Availability** Software is available in the web. Description is in the text.

## Declarations

**Conflict of interest** Not applicable.

**Ethical approval and consent to participate** Not applicable.

**Consent for publication** All authors agreed with the content of this paper.



## References

1. Sugeno, M.: Theory of Fuzzy Integrals and Its Applications. Tokyo Institute of Technology,, Ph.D. Thesis (1974)
2. Torra, V., Narukawa, Y., Sugeno, M.: Non-Additive Measures, Theory and Applications, Studies in Fuzziness and Soft Computing. Springer, Berlin (2013). https://doi.org/10.1007/978-3-319-03155-2
3. Dimuro, P.G., Fernandez, J., Bedregal, B., Mesiar, R., Sanz, J.A., Lucca, G., Bustince, H.: The state-of-art of the generalizations of the choquet integral: From aggregation and pre-aggregation to ordered directionally monotone functions. Inf. Sci. **57**, 27–43 (2020). https://doi.org/10.1016/j.inffus.2019.10.005

4. Torra, V.: On a family of fuzzy measures for data fusion with reduced complexity. Proc. 3rd Int. Conf on Information Fusion, TuCa17-23 (2000). https://doi.org/10.1109/IFIC.2000.862689
5. Saleh, E., Valls, A., Moreno, A., Romero-Aroca, P., Bustince, H., Torra, V.: A hierarchically ⊥-decomposable fuzzy measure-based approach for fuzzy rules aggregation. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **27**, 59–76 (2019). https://doi.org/10.1142/S0218488519400038
6. Marco-Detchart, C., Lucca, G., Lopez-Molina, C., De Miguel, L., Dimuro, P.G., Bustince, H.: Neuro-inspired edge feature fusion using choquet integrals. Inf. Sci. **581**, 740–754 (2021). https://doi.org/10.1016/j.ins.2021.10.016
7. Beliakov, G.: Construction of aggregation functions from data using linear programming. Fuzzy Sets and Systems **160**, 65–75 (2009). https://doi.org/10.1016/j.fss.2008.07.004
8. Javier, M., Serge, G., Pilar, B.: k-maxitive fuzzy measures: A scalable approach to model interactions. Fuzzy Sets and Systems **324**, 33–48 (2017). https://doi.org/10.1016/j.fss.2017.04.011
9. Beliakov, G., Wu, J.Z.: Learning fuzzy measures from data: Simplifications and optimisation strategies. Information Sciences **494**, 100–113 (2019). https://doi.org/10.1016/j.ins.2019.04.042
10. Grabisch, M.: New algorithm for identifying fuzzy measures and its application to pattern recognition. IEEE International Conference on Fuzzy Systems (1995)
11. Kojadinovic, I.: Minimum variance capacity identification. European Journal of Operational Research **177**, 498–514 (2007). https://doi.org/10.1016/j.ejor.2005.10.059
12. Combarro, E.F.: Identification of fuzzy measures from sample data with genetic algorithms. Computers & Operations Research **33**, 3046–3066 (2006). https://doi.org/10.1016/j.cor.2005.02.034
13. Grabisch, M., Marichal, J.L., Mesiar, R.: Aggregation Functions, Encyclopedia of Mathematics and Its Applications. Cambridge University Pres, ??? (2009). https://doi.org/10.1017/CBO9781139644150
14. Grabisch, M., Kojadinovic, I., Meyer, P.: A review of methods for capacity identification in choquet integral based multi-attribute utility theory: Applications of the kappalab r package. European Journal of Operational Research **186**, 766–785 (2008). https://doi.org/10.1016/j.ejor.2007.02.025
15. Grabisch, M., Kojadinovic, I., Meyer, P.: Non-Additive Measure and Integral Manipulation Functions, (kappalab package in R). https://CRAN.R-project.org/package=kappalab (2015)
16. Rota, G.C.: On the foundations of combinatorial theory. i. the theory of möbius functions. Z. Wahrscheinlichkeitstheorie verw Gebiete **2**, 340–368 (1964). https://doi.org/10.1007/BF00531932
17. Grabisch, M.: The application of fuzzy integrals in multi criteria decision making. European Journal of Operational Research **89**, 445–456 (1996). https://doi.org/10.1016/0377-2217(95)00176-X
18. Mesiar, R.: Generalizations of *k*-order additive discrete fuzzy measures. Fuzzy Sets and Systems **102**, 423–428 (1999). https://doi.org/10.1142/S0218488599000489
19. Grabisch, M.: *k*-order additive discrete fuzzy measures and their representation. Fuzzy Sets and Systems **92**, 167–189 (1997). https://doi.org/10.1016/S0165-0114(97)00168-1
20. Choquet, G.: Theory of capacities. Annales de L'Institut Fourier **5**, 131–295 (1954). https://doi.org/10.5802/aif.53
21. Imai, D. H. Asano, Sato, Y.:An algorithm based on alternative projections for a fuzzy measure identification problem. In:Torra, V. (ed.) Information Fusion in Data Mining. Studies in Fuzziness and Soft Computing, pp. 149–158. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/978-3-540-72434-6
22. Torra, V., Narukawa, Y.: Modeling Decisions: Information Fusion and Aggregation Operators. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-68791-7
23. http://www.mdai.cat/ifao/