A New Algorithm for Identifying Fuzzy Measures and Its Application to Pattern Recognition

Michel GRABISCH
Thomson-CSF, Central Research Laboratory
Domaine de Corbeville, Orsay, France
email grabisch@thomson-lcr.fr

Abstract

We present a new algorithm for identifying fuzzy measures, which is a kind of gradient algorithm with constraints. Its performance is superior to the one of previous attempts, and we show its efficiency into a problem of pattern recognition using Choquet integral.

1 Motivations

Since Sugeno proposed the concept of fuzzy measure and fuzzy integral in 1974, the theory has been well developed, in the fuzzy community (essentially by Weber, Wang, Murofushi, De Campos) as well as in utility theory, where fuzzy measures are called non-additive measures (essentially the works of Schmeidler, Gilboa, Quiggin) and other fields of decision theory, as game theory (Shapley and Aumann). Some recent books give an overview of the subject under these different points of view, as the one of Wang and Klir [12] (the Sugeno integral), Sugeno and Murofushi [11] (general), Nguyen *et al.* [9] (decision theory), Quiggin [10] (utility theory), and Denneberg [1] (mathematics).

As remarked by the author [4], most real applications of fuzzy measures deal with multicriteria decision problems, where fuzzy measures are defined on the (finite) set of criteria or attributes, and model the relative importance of criteria as well as their interaction. Indeed, modeling of interaction among criteria is precisely the main interest of fuzzy measures and integrals (see e.g. [6] for a study of the use of fuzzy measures and integrals in the field of multicriteria decision making).

Despite the success of such techniques in real applications, the practical use of fuzzy measures could be difficult, because for a n criteria decision problem, one has to identify 2^n coefficients in order to define a fuzzy measure. This identification step is very important since all the knowledge concerning the criteria is embedded into the fuzzy measure. At this stage, several alternatives exist, which are described in [9]:

- use of an expert to assess the 2^n coefficients, on the basis of semantical considerations, which are issued mainly from utility theory,
- use of standard optimization algorithms, as the Lemke's method, constrained least mean squares, etc
- a mix of the two above

Clearly, the first alternative is of practical use only for low values of n. On the other hand, standard optimization algorithms are rather greedy, and are not suitable to fuzzy measures since they lead to sparse matrices of constraints. This is due to the peculiar structure of the fuzzy measure coefficients, which form a lattice. The third alternative seems to be the best, but here again arises the problem of a *suitable* optimization algorithm. In this paper, we precisely intend to solve this point, that is, to propose a new algorithm for identifying fuzzy measures, which takes advantage of the lattice structure of the coefficients. In fact, some attempts in this direction already exist, as the algorithm of Mori and Murofushi [7, 9]. We will see that our algorithm provides better performance.

The paper is organized as follows: section 2 gives necessary definitions on fuzzy measures and integrals, illustrated with some properties. Section 3 describes the algorithm together with its philosophy, section 4 reports some experiments to compare with previous attempts, and section 5 gives a complete application of the algorithm in the field of pattern recognition. Finally, section 6 concludes the paper.

2 Background on fuzzy measures and integrals

We give now essential definitions, restricted to the finite case as explained above. In all this paper, min and max are denoted $\wedge, \vee$ respectively.

We consider a finite universe $X = \{x_1, \ldots, x_n\}$.

Definition 1 *A* fuzzy measure μ *defined on* X *is a set function* $\mu : \mathcal{P}(X) \longrightarrow [0,1]$ *verifying the following axioms:*

(i) $\mu(\emptyset) = 0, \mu(X) = 1.$

(ii) $A \subseteq B \Rightarrow \mu(A) \leq \mu(B)$

$\mathcal{P}(X)$ indicates the power set of X, i.e. the set of all subsets of X.

Note that the usual additivity axiom for probability measures $\mu(A \cup B) = \mu(A) + \mu(B), A \cap B = \emptyset$, has been replaced by a weaker one: monotonicity. For this reason, the definition of a fuzzy measure requires in general 2^n coefficients, namely the measures of the 2^n subsets of X. Fuzzy measures include as particular cases probability measures, possibility and necessity measures, belief and plausibility functions, etc.

A convenient way of representing fuzzy measures in the finite case is to use a lattice representation. All the 2^n coefficients defining a fuzzy measure can be arranged in a lattice with the usual ordering on real numbers, which is in fact the same as the Boolean lattice of subsets of X, ordered with inclusion. Figure 1 gives an illustration when $n = 4$ (for the sake of simplicity, μ_{23} denotes $\mu(\{x_2, x_3\})$ and similarly for all coefficients).

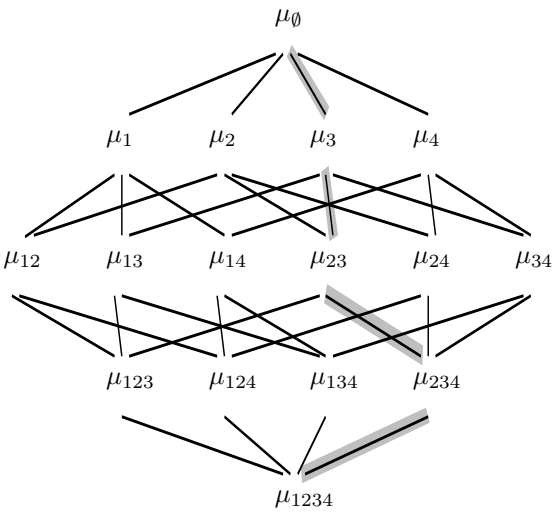

Figure 1: Lattice of the coefficients of a fuzzy measure ($n = 4$)

At this stage, we introduce some vocabulary about this lattice, which will be used in the sequel. The lattice of fuzzy measure is made from *nodes* related by *links*. The lattice has $n + 1$ horizontal *layers*, numbered from 0 (for the layer containing only $\mu_\emptyset$) to n (for the layer containing only μ_X). A *path* is a set of chained links, starting from the node $\mu_\emptyset$ and arriving to the node u_X (on figure 1, the path passing through μ_3, μ_{23}, and μ_{234} is emphasized). For a given node in layer l, its *lower neighbors* (resp. *upper neighbors*) are the set of nodes in the layer $l-1$ (resp. $l+1$) linked to it. There are l lower neighbors and $n-l$ upper neighbors.

We turn now to fuzzy integrals, that is, integrals of a real function with respect to a fuzzy measure, by analogy with Lebesgue integral which is defined with respect to an ordinary (i.e. additive) measure. Sugeno has proposed the following definition, restricted to functions on $[0, 1]$:

Definition 2 *Let μ be a fuzzy measure on X. The* discrete Sugeno integral *of a function $f : X \longrightarrow [0, 1]$ with respect to μ is defined by :*

$$\mathcal{S}_\mu(f(x_1), \ldots, f(x_n)) \triangleq \bigvee_{i=1}^{n} (f(x_{(i)}) \wedge \mu(A_{(i)})) \quad (1)$$

where $\cdot_{(i)}$ indicates that the indices have been permuted so that $0 \leq f(x_{(1)}) \leq \cdots \leq f(x_{(n)}) \leq 1$, and $A_{(i)} \triangleq \{x_{(i)}, \ldots, x_{(n)}\}$.

Another definition is due originally to Choquet, and can be applied to any positive function:

Definition 3 *Let μ be a fuzzy measure on X. The* discrete Choquet integral *of a function $f : X \longrightarrow \mathbb{R}^+$ with respect to μ is defined by*

$$\mathcal{C}_\mu(f(x_1), \ldots, f(x_n)) \triangleq \sum_{i=1}^{n} (f(x_{(i)}) - f(x_{(i-1)}))\mu(A_{(i)}) \quad (2)$$

with the same notations as above, and $f(x_{(0)}) = 0$.

The definition can be extended to negative functions too (see e.g. [1, 9]). The main advantage of Choquet integral is that it coincides with Lebesgue integral when the measure is additive. More general definitions including Sugeno and Choquet definitions exist, but will not be considered here, we refer the reader to [2, 8, 9]. In the sequel, the use of $\mathcal{F}_\mu$ in a formula will indicate that either $\mathcal{S}_\mu$ or $\mathcal{C}_\mu$ can be used.

An important property of fuzzy integral which will be used in the sequel is the following:

Property 1 *The values of the Sugeno integral and the Choquet integral on the 2^n vertices of the hypercube $[0, 1]^n$ give the 2^n coefficients of the fuzzy measure μ:*

$$\mathcal{S}_\mu(\delta_1, \delta_2, \ldots, \delta_n) = \mu(\bigcup_{i|\delta_i=1} \{x_i\}), \quad \forall(\delta_1, \ldots, \delta_n) \in \{0, 1\}^n$$

$$\mathcal{C}_\mu(\delta_1, \delta_2, \ldots, \delta_n) = \mu(\bigcup_{i|\delta_i=1} \{x_i\}), \quad \forall(\delta_1, \ldots, \delta_n) \in \{0, 1\}^n$$

i.e. for example $\mathcal{S}_\mu(0, 1, 0, \ldots, 0) = \mu(\{x_2\})$.

3 Description of the algorithm

Suppose we have a system $\mathcal{S}$ with n inputs variables and one output we want to model under the form of a fuzzy integral, that is

$$\mathcal{S}(x) = \mathcal{F}_\mu(x) \quad (3)$$

where $x = [x_1 \cdots x_n]$ is the n-dimensional input vector, and μ a fuzzy measure on the set X of input variables. Looking at the notations of the previous paragraph, one can see we have kept the same notation for variables in X

and values of variables, but this should cause no confusion. Having a set of learning data (x, y) where y denotes the output of the system when the input is x, the problem is to find for a given fuzzy integral (Choquet, Sugeno, or other) the best fuzzy measure μ so that an error criterion such as the sum of squared errors between the model and the system is minimized.

An analysis of the problem reveals the following:

- one of the major difficulty of the problem is that the coefficients of the fuzzy measure must obey monotonicity constraints, which can be put under a lattice form.
- whatever the kind of integral is chosen, an input x involves the use of all the coefficients situated on a path from $\emptyset$ to X. The particular path used depends only on the ordering of the values $x_1, \ldots, x_n$ (see figure 1, where the path corresponding to a datum such that $x_1 \leq x_4 \leq x_2 \leq x_3$ is emphasized).
- if too few data are used, then it may be that some coefficients of the lattice are not used, so that they cannot be modified by some gradient considerations. In fact, it can be shown [5, 9] that at least $n!/[(n/2)!]^2$ (for n even), or $n!/([(n-1)/2]![(n+1)/2]!)$ (for n odd) data are necessary to use all the coefficients.
- it is known that fuzzy integrals range between *min* and *max*, that is our model is suitable for systems such that for any input vector, $x_1 \wedge \cdots \wedge x_n \leq \mathcal{S}(x_1, \ldots, x_n) \leq x_1 \vee \cdots \vee x_n$. In the absence of any learning data and any information, the only reasonable solution seems to be to choose the average value of the input $1/n \sum_i x_i$. This corresponds to a Choquet integral with respect to a fuzzy measure which is additive and equidistributed, i.e. $\mu(\{x_i\}) = 1/n$, for all i. In the lattice representation, this corresponds to a state where every node in a layer l is equidistant from any node of the layer $l+1$ or $l-1$. For this reason, we call this the *equilibrium state of the fuzzy measure*. It should be reasonable to get fuzzy measures as near as possible to this equilibrium state.

These considerations leads to an algorithm which is based on two fundamental steps:

step 1: for a given datum x , we modify only the coefficients on the path involved by x in order to decrease the error, as in a gradient descent algorithm. The modification is done in order to preserve the monotonicity property *on the path*. Also, monotonicity is checked for neighboring nodes. This is done for all the learning data, several times.

step 2: if there are too few learning data, then some nodes may have been left unmodified. These nodes are modified here in order to have the most equilibrated lattice, i.e. distance from neighbors should be as equal as possible.

The originality of our algorithm is more in step 2, since Mori and Murofushi [7] have already proposed an heuristic algorithm which is similar to our step 1. The main idea behind step 2 is the following: in the absence of any information for some nodes, we should arrange them into the lattice in order to get a lattice as homogeneous as possible. We expect by this to be robust when only few data are available. Using the language of pattern recognition, we could say that we expect better generalization ability, or less overtraining.

We detail now the two steps:

step 0 : the fuzzy measure is initialized at the equilibrium state.

step 1.1 : consider a learning datum (x, y). Compute the model error $e = \mathcal{F}_\mu(x) - y$. Let us denote $u(0), u(1), \ldots, u(n)$ the values of the nodes on the path involved by x. For example, in figure 1, we have $u(1) = \mu_3$, $u(2) = \mu_{23}$, $u(3) = \mu_{234}$. Note that we have always $u(0) = 0$, $u(n) = 1$.

step 1.2 : we compute the new value $u^{\text{new}}(i)$ as follows:

$$u^{\text{new}}(i) = u^{\text{old}}(i) - \alpha \frac{e}{e_{\max}} (x_{(n-i)} - x_{(n-i-1)}) \quad (4)$$

$\alpha \in [0, 1]$ is a constant[1], and $e_{\max}$ is the maximum value of the error. $e_{\max} = 1$ if y takes its values in $[0, 1]$. As before, $x_{(i)}$ indicates the ith value of the x_i in ascending order.

step 1.3 : verify the monotonicity relations. If $e > 0$, the verification is done for lower neighbors only, and if $e < 0$, for upper neighbors only. Moreover, the verification is done only for nodes *already modified* in previous steps. If a monotonicity relation is violated, say with node μ_J, $J \subset X$, then $u(i) = \mu_J$.

Repeat steps 1.2 and 1.3 for $i = 1, \ldots, n-1$, in the following order:

- if $e > 0$, we begin by $u(1), u(2), \ldots, u(n-1)$
- if $e < 0$, we begin by $u(n-1), u(n-2), \ldots, u(1)$

Repeat steps 1.1 to 1.3 for all learning data. This is called *one iteration*. Several iterations can be performed, i.e. the same set of data is used several times.

step 2.1 : for every node left unmodifed in step 1 (begin by lower levels), verify monotonicity relations with upper and lower neighbors. If they are not verified, modify the node as in step 1.3. It may happen here that upper and lower neighbors do not themselves satisfy the monotonicity relations. In this case, we correct at first the nodes violating monotonicity.

[1] Also, α can be a value decreasing at each iteration.

step 2.2 : for every node left unmodified in step 1 (begin by lower levels), adjust its value considering the values of its upper and lower neighbors, in order to have an homogeneous lattice. This is done by computing the following quantities. Let $u(i)$ be the node considered:

- mean value of upper neighbors $\overline{m}(i) = 1/(n-i)\sum_{\text{upper neighbors}} \mu_J$
- mean value of lower neighbors $\underline{m}(i) = 1/i \sum_{\text{lower neighbors}} \mu_J$
- minimum distance between $u(i)$ and its upper (resp. lower) neighbors, denoted $\overline{d}_{\min}(i)$, (resp. $\underline{d}_{\min}(i)$

If $\overline{m}(i) + \underline{m}(i) - 2u(i) > 0$, then $u(i)$ is increased

$$u^{\text{new}}(i) = u^{\text{old}}(i) + \beta \frac{(\overline{m}(i) + \underline{m}(i) - 2u(i))\overline{d}_{\min}(i)}{2(\overline{m}(i) + \underline{m}(i))} \quad (5)$$

otherwise $u(i)$ is decreased

$$u^{\text{new}}(i) = u^{\text{old}}(i) + \beta \frac{(\overline{m}(i) + \underline{m}(i) - 2u(i))\underline{d}_{\min}(i)}{2(\overline{m}(i) + \underline{m}(i))} \quad (6)$$

β is a constant value in $[0,1]$ (as before, β can be made decreasing at each iteration).

Do steps 2.1. and 2.2 for every node left unmodified in the first step. This is called one iteration, and several iterations can be done.

4 Properties and tests

The following properties can be noticed:

- when a datum has passed steps 1.2 and 1.3, the monotonicity on the path is ensured, as well as the monotonicity with neighbouring already modified nodes.
- for the Choquet integral, clearly equation (4) corresponds to the gradient descent algorithm of criterion $E = (\mathcal{C}_\mu(x) - y)^2$, with learning rate $\alpha/2e_{\max}$.
- if the same datum is entered repeatedly, then the algorithm converges.
  (sketch of the proof: consider a datum (x, y) with $y = \mathcal{C}_\mu(x)$ for simplicity. Let us denote $u^k(i)$ the value of node $u(i)$ at the kth iteration, and $u(i)$ the true value. Then we have:
  $u^1(i) = u^0(i) - \alpha(\mathcal{C}_{\mu^0}(x) - \mathcal{C}_\mu(x))(x_{(n-i)} - x_{(n-i-1)})$. Developping and considering only the terms in $u^k(i)$, we have $u^1(i) = Ku^0(i) + (1-K)u(i)$, with $K = 1 - \alpha(x_{(n-i)} - x_{(n-i-1)})^2$, so that $K \in [0,1]$. Then it is easy to prove that $u^k(i) = K^k u^0(i) + (1-K^k)u(i)$, so that $u^k(i)$ converges to the true value $u(i)$.)
- if $\alpha = 1$, and if we feed the algorithm with learning data corresponding to the 2^n vertices of the hypercube $[0,1]^n$, then the algorithm converges exactly to the solution in one iteration.
  (sketch of the proof: from property 1, clearly the y are the 2^n coefficients of the fuzzy measure, so that we have:
  $u^1(i) = u^0(i) - (\mathcal{C}_{\mu^0}(x) - u(i)) = u(i)$.)

We now give some results of test in order to compare our algorithm with the one of Mori and Murofushi. They proposed an algorithm similar to our step 1, where the updating formula for a datum (x, y) is:

$$u^{\text{new}}(i) = u^{\text{old}}(i) - KC_n^i \frac{e}{x_{(n)}}$$

The test consists of identifying the following fuzzy measure μ (table 1), using 81 samples data (x, y), with $y = \mathcal{C}_\mu(x) + n$, where n is a centered gaussian noise, and x is any input point in $[0,1]^4$ whose coordinates belong to the set $\{0., 0.5, 1.\}$. The result of identification for different

A	$\mu(A)$	A	$\mu(A)$	A	$\mu(A)$
$\{1\}$	0.1	$\{1,2\}$	0.3	$\{1,2,3\}$	0.5
$\{2\}$	0.2105	$\{1,3\}$	0.3235	$\{1,2,4\}$	0.8667
$\{3\}$	0.2353	$\{1,4\}$	0.7333	$\{1,3,4\}$	0.8824
$\{4\}$	0.6667	$\{2,3\}$	0.4211	$\{2,3,4\}$	0.9474
		$\{2,4\}$	0.8070		
		$\{3,4\}$	0.8235		

Table 1: Fuzzy measure to be identified

values of variance of the noise is given in the comparative table 2, where $E^2 = 1/81 \sum (y - \mathcal{C}_\mu(x))^2$. Performance

σ^2	Mori & Murofushi		our algorithm	
noise	n. iter.	E^2	n. iter.	E^2
0.0	97	0.0000	8	1.4e-7
0.00096	116	0.00087	10	0.00083
0.0125	70	0.0117	11	0.0108
0.00625	74	0.0605	11	0.0530
0.01250	79	0.1211	9	0.1054

Table 2: Results of identification for the two algorithms

is clearly better for our algorithm, especially in term of speed of convergence. The experiment has been realized with $\alpha = 0.5$, and the number of iterations indicated corresponds to a stability of E^2 at 10^{-6}. The number of iterations has been extended to 100 to verify the stability of the result. With a lower value of α, precision is increased, at the price of a longer time of convergence. This is summarized in table 3, when $\sigma^2 = 0.0125$.

We give now a comparison with a standard optimization algorithm, namely the Lemke's method (quadratic constrained programming method), which we perform on the

α	n. iter.	E^2
0.5	11	0.0107846
0.25	27	0.00964529
0.1	52	0.00899498
0.05	86	0.00878478

Table 3: Effect of parameter α

tiles data of Zimmermann [14]. The tiles data are a set of 24 data with $n = 2$. The Lemke's method gives the optimal unique solution to the minimization of the total squared error $\sum(y - \mathcal{C}_\mu(x))^2$. If we compare it to the solution provided by our algorithm, taking $\alpha = 0.05$ (see table 4), one can see that the two results are almost similar. This shows that our algorithm is nearly optimal.

method	total squared error	$\mu(\{x_1\})$	$\mu(\{x_2\})$
Lemke	0.0602	0.27762	0.385163
ours	0.0613	0.281089	0.383663

Table 4: Comparison on the tiles data set

All the previous experiments concerned the step 1, since the number of data was sufficient. Now we turn to the examination of step 2. As we said before, the role of step 2 is to get a fuzzy measure as homogeneous as possible. In order to verify this, we feed step 2 with a fuzzy measure which has been distorded in the following way:

- let μ be the additive equidistributed measure (equilibrium state), with $n = 4$.
- distord measures of subsets on the following path towards low values: $\mu_3 = 0$, $\mu_{23} = 0.2$, and $\mu_{234} = 0.4$.
- conversely, distord the nodes of another path towards high values: $\mu_1 = 0.6$, $\mu_{13} = 0.9$, and $\mu_{123} = 1$.

Note that the fuzzy measure μ is no more monotonic. If we suppose that the above modified nodes are the result of step 1, we expect as a result of step 2 the following:

- the measure is monotonic
- nodes which are in the vicinity of path 1-13-123 (i.e. μ_{12}, μ_{14}, and μ_{134} for the immediate neighbors) are more or less between equilibrium state and high values of this path.
- similarly, nodes in the vicinity of path 3-23-234 (i.e. μ_{34}) are between the equilibrium state and low values.

The result of step 2 after 10 iterations with $\beta = 1$ is given in table 5. Figures in bold face indicates fixed values (imposed by step 1). It can be seen that the result satisfies the requirements.

A	$\mu(A)$	A	$\mu(A)$	A	$\mu(A)$
$\{1\}$	**0.6**	$\{1,2\}$	0.642	$\{1,2,3\}$	**1.0**
$\{2\}$	0.2	$\{1,3\}$	**0.9**	$\{1,2,4\}$	0.778
$\{3\}$	**0.0**	$\{1,4\}$	0.627	$\{1,3,4\}$	0.9
$\{4\}$	0.235	$\{2,3\}$	**0.2**	$\{2,3,4\}$	**0.4**
		$\{2,4\}$	0.4		
		$\{3,4\}$	0.383		

Table 5: Result of step 2

5 Application to pattern recognition

We illustrate the use of the algorithm in pattern recognition. The author has previously proposed a method of classification based on fuzzy integral (see [3, 5] for details). Let us consider the case of two classes for simplicity. Roughly speaking, the method consists of combining the decision of several sensors by Choquet integral, the fuzzy measure expressing the relative importance of sensors and their interaction for a given class. The crucial point in this method is to identify correctly the fuzzy measures, using training data. It has been shown that the best criterion for doing this is the following:

$$E^2 = \sum_j \sum_k |\Psi(\Delta\Phi_{12}(X_k^j)) - 1|^2$$

where X_k^j is the kth training data of class j, Ψ a sigmoid-type function, and $\Delta\Phi_{12}$ is given by:

$$\Delta\Phi_{12}(X_k^j) = \Phi_{\mu_1}(C_1|X_k^j) - \Phi_{\mu_2}(C_2|X_k^j)$$

for $j = 1$, and 1 and 2 being inverted for $j = 2$. $\Phi_{\mu_i}(C_i|X_k^j)$ is the Choquet integral w.r.t fuzzy measure μ_i, combining the degrees of certainty that X_k^j belongs to class C_i. Until now, the minimization of J required a constrained least mean squares (CLMS) algorithm, which although efficient, is time and memory consuming. Moreover, CLMS requires a sufficient number of data.

Keeping the same criterion J, we use instead of CLMS our algorithm, with slight adaptation. The expression of e in formula (4) is now $\Psi(\Delta\Phi_{12}(X_k^j)) - 1$, which is always ≤ 0. Thus, for a training datum belonging to class 1, we increase μ_1 using (4) and decrease μ_2 using (4) but with $-e$ instead of e. We call this algorithm Heuristic Least Mean Squares (HLMS) in the sequel.

We give now some experimental results, on real and simulated data. As real data, we used the cancer data (2 classes, 9 attributes, 284 data), and for simulated data, we used a 2 classes, 5 attributes non gaussian problem, whose classes are very intricated, with 200 data.

- comparison of CPU time, and final value of criterion J. Since the criterion is the same for CLMS and HLMS, we can compare their final value, in order to see how close to optimal solution HLMS can

come. The test has been done on the whole set of cancer data, after a reduction to 7 attributes by principal component analysis (results on table 6). The initial value of J is 237.22. As it can be seen, the

	CPU (s)	final value of J	learning rate (%)
CLMS	1054	150.44	84.5
HLMS	5	167.	83.8

Table 6: Comparison of CLMS and HLMS (1)

ratio of CPU time is impressive, for a slight loss in optimality.

- effect of a few number of training data. Taking the cancer data, CLMS cannot be performed without a reduction of the number of attributes to 7 (too few data w.r.t. to the number of attributes). But HMLS does not have any restriction on the number of data. A comparison of the respective classification rates is given in table 7, where the rate was estimated by a random subsampling with 70%, 10 runs (standard deviation of estimate is 2.79%). As it can be seen,

	classification rate (%)
CLMS	72.9
HLMS	77.4
HLMS without step 2	76.9

Table 7: Comparison of CLMS and HLMS (2)

the improvement is significant. Moreover, as far as the author knows, the best performance recorded for the cancer data was 77.1%, by Predictive Value Maximization Rule [13], implying that fuzzy integral achieves the best known result. Note the slight loss in performance when step 2 of the algorithm is not performed.

Another test was performed with the simulated data, taking only 20% of the data for training (this is the lower limit for CLMS). The comparison is given in table 8, where the classification rate has been estimated by performing 100 runs of a random subsampling at 20%.

	classification rate (%)
CLMS	69.8
HLMS	71.4

Table 8: Comparison of CLMS and HLMS (3)

Clearly, HLMS performs better in the situations tested above. Also, it can be used even for a high number of attributes, since it does not require much memory to work. For more standard situations, i.e. few attributes, sufficient training data, CLMS performs better than HLMS, since the latter is suboptimal, but at the price of a much higher CPU time.

6 Concluding remarks

We have proposed a new algorithm of identification of fuzzy measures, and shown its efficiency through several tests in modeling and pattern recognition. Although a deeper study of the algorithm seems to be necessary, in particular concerning convergence properties, monotonicity, etc, the first results presented here are quite encouraging. A next topic of study would be the use of this algorithm for other kinds of fuzzy integrals, as Sugeno integral and fuzzy t-conorm integrals.

References

[1] D. Denneberg, *Non-Additive Measure and Integral*, Kluwer Academic, 1994.

[2] M. Grabisch, T. Murofushi, M. Sugeno, Fuzzy Measure of Fuzzy Events Defined by Fuzzy Integrals, *Fuzzy Sets & Systems* 50 (1992), 293-313.

[3] M. Grabisch, M. Sugeno, Multi-attribute classification using fuzzy integral, *1st FUZZ'IEEE Congress*, San Diego, march 92, 47-54.

[4] M. Grabisch, A survey of applications of fuzzy measures and integrals, *5th Int. Fuzzy Systems Assoc. Conf.*, Seoul, Korea, july 1993.

[5] M. Grabisch. J.M. Nicolas, Classification by fuzzy integral — Performance and tests, to appear in *Fuzzy Sets & Systems*, special issue on Pattern Recognition.

[6] M. Grabisch, Fuzzy integral in multicriteria decision making, to appear in *Fuzzy Sets & Systems, Special issue on 5th IFSA Congress*

[7] T. Mori, T. Murofushi, An analysis of evaluation model using fuzzy measure and the Choquet integral, *5th Fuzzy Systems Symposium*, Kōbe, Japan, june 2-3 1989 (in japanese).

[8] T. Murofushi, M. Sugeno, Fuzzy t-conorm integrals with respect to fuzzy measures: generalization of Sugeno integral and Choquet integral, *Fuzzy Sets & Systems* 42 (1991), 57-71.

[9] H.T. Nguyen, E.A. Walker, M. Grabisch, *Fundamentals of Uncertainty Calculi, with Applications to Fuzzy Inference*, Kluwer Acad., to appear in 1994.

[10] J. Quiggin, *Generalized Expected Utility Theory*, Kluwer Academic, 1993.

[11] M. Sugeno, T. Murofushi, *Fuzzy Measure*, Course in Fuzzy Theory, Vol. 3, Nikkan Kōgyō Shimbunsha, 1992 (in japanese).

[12] Z. Wang, G. Klir, *Fuzzy Measure Theory*, Plenum Press, 1992.

[13] S.M. Weiss, I. Kapouleas, An empirical comparison of pattern recognition, neural nets, and machine learning classification methods, *Proc. 11th IJCAI*, 781-787.

[14] H.-J. Zimmermann, P. Zysno, Latent Connectives in Human Decision Making, *Fuzzy Sets & Systems* 4 (1980), 37-51.