KYBERNETIKA — VOLUME 40 (2004), NUMBER 1, PAGES 51 - 70

NONPARAMETRIC RECURSIVE AGGREGATION PROCESS

ELENA TSIPORKOVA AND VESELKA BOEVA

In this work we introduce a nonparametric recursive aggregation process called Multilayer Aggregation (MLA). The name refers to the fact that at each step the results from the previous one are aggregated and thus, before the final result is derived, the initial values are subjected to several layers of aggregation. Most of the conventional aggregation operators, as for instance weighted mean, combine numerical values according to a vector of weights (parameters). Alternatively, the MLA operators apply recursively over the input values a vector of aggregation operators. Consequently, a sort of unsupervised self-tuning aggregation process is induced combining the individual values in a certain fashion determined by the choice of aggregation operators.

Keywords: multilayer aggregation operators, power means, monotonicity
AMS Subject Classification: 26E60, 47A64, 26A48, 47B99

1. INTRODUCTION

Aggregation of information in general, is occurring in one or another form in almost any theoretical or practical domain nowadays. Practically, any time one piece of knowledge, data, quantity etc. is derived from a set of multiple ones, an aggregation has been performed. An adequate aggregation process is expected to combine multiple values into a single one, so that the final result of aggregation reflects in some fashion all the individual values. The scientific literature is abundant of numerous examples of aggregation constructions, each one claiming to offer a more fair treatment and combination of the individual values for aggregation than the rest. However, aggregation is no more only a pure theoretical topic, meant solely to deliver nice mathematical papers. Various software applications nowadays require development and implementation of effective aggregation procedures. This has inspired a search for new aggregation methods answering some typical software application requirements, as for instance, allowing for an algorithmization and application independent definition. The latter expresses the fact that the aggregation process will not depend on numerous different parameters that have to be initialized, trained and tuned for any particular application domain.

The aggregation algorithm considered herein, has been inspired by and verified on a concrete aggregation problem occurring in speech recognition applications. Most speech recognizers usually assign to each output word some confidence score expressing the degree of certainty that the word in question has really been uttered by the user. All these word confidences are combined in a sentence confidence score enabling the speech recognition application to detect and reject any intruder sentence, i. e. one that does not belong to the particular application vocabulary. Naturally, the sentence confidence scores are expected to take into account, in a suitable fashion, all the individual word confidences. The choice of aggregation operator is therefore crucial. Some aggregation operators can lead to a significant loss of information since their values can be greatly influenced by extreme scores, while others are penalizing too much for low-scoring outliners. One possible solution is to select the best operator via some learning procedure. However, there are several problems associated with such an approach. On the first place, learning requires availability of application data, which is not always possible in the speech recognition domain. The enormous variability of applications (telephone numbers, names, games, etc.), languages (English, French, Spanish, etc.) and acoustic environments (telephone, car, office, etc.) makes data collection for speech recognition very complex and expensive process. Secondly, even one has data to learn the best aggregation operator, there is no doubt that this operator will be different for different applications, as for example an operator that is good for telephone number recognition task cannot be expected to suit name recognition task. However one cannot afford to build different speech recognizers for different applications. In general, the pragmatic approach is to use one speech engine for a given acoustic environment with parameters tuned in such a way that they serve as an optimal trade-off for a wide set of different applications.

Another possible and rather straightforward solution to the described problem is to use different aggregation operators in order to find some trade-off between their conflicting behaviour. In this way different aspects of the input values will be taken into account in the process of aggregation. Various approaches for combining aggregation operators can be found in the literature. For instance, a convex combination of two aggregation operators A_1 and A_2 has been exploited in several different ways: $A_1^{\lambda}(x,y)A_2^{1-\lambda}(x,y)$ (cf. [12]), $\lambda A_1(x,y) + (1-\lambda)A_2(x,y)$ (cf. [5], [9]), and $A_1(A_2(x,\lambda), A_2(y,1-\lambda))$ (see [6]), for $\lambda \in [0,1]$. Next in [4], it is proposed to take some aggregation operators $A_1, A_2, \ldots, A_m, A_{m+1}$ and then to construct a new aggregation operator as follows $A_{m+1}(A_1, A_2, \ldots, A_m)$.

Our approach, though much more applied, follows the idea, proposed by Matkowski in [2, 3]. He considers $A_1, A_2, \ldots, A_m$ continuous means[1] and defines the functions $A_{i,n}$, $i = 1, \ldots, m$ and $n = 1, 2, \ldots$, as follows:

$$A_{i,1} = A_i \quad i = 1, \ldots, m$$

$$A_{i,n+1}(x_1, \ldots, x_m) = A_i(A_{1,n}(x_1, \ldots, x_m), \ldots, A_{m,n}(x_1, \ldots, x_m)).$$

Similarly we suggested in [8] an iterative aggregation with aggregation operators

[1]Means are actually nothing more but compensatory aggregation operators as defined in the following section.

$A_1, A_2, \ldots, A_m$ until some stop condition is satisfied. Thus the final result is conceived after passing a few layers of aggregation. At the first layer, we have the list of initial values that are to be combined. Using a vector of aggregation operators new values are obtained and if the stop condition is not fulfilled then the next step is to combine these new values again using the given aggregation operators. This process is repeated again and again until the stop condition is satisfied. Consequently, it will be referred to as Multilayer Aggregation (MLA). This MLA algorithm is interesting for the speech recognition problem considered above in two ways: (i) it is a cheap solution since it does not require training data for off-line parameter learning; (ii) it offers a more universal approach for different applications by acting as a trade-off between aggregation operators that are good for one application, but damaging for another and vice versa.

2. MULTILAYER AGGREGATION

2.1. Motivation

Let us motivate our new aggregation algorithm with a particular speech recognition example. In general, any speech recognition application is designed for a particular application domain in mind. For instance, consider a digit string recognition task with an underlying application grammar allowing for a recognition of any length sequence of the digits between 0 and 9. The appendix presents output data for 11 sentences from a recognition session on digits strings in French. The sentences are classified in two main groups: correct recognitions and recognition errors. The former refers to a correct user input (sentences 1–3 and 11), i. e. a 0–9 digit strings, that have been unmistakably determined by the recognizer. The latter summarizes two rather different cases: (i) a correct user input that for one or another reason has been misrecognized (see sentences 4 and 6 where "sept" and "deux" have been erroneously recognized as "cinq" and "neuf" respectively, and an extra "huit" has been added at the end of sentence 4); (ii) a user input containing words that do not belong to the application dictionary, i. e. only digits between 0 and 9 (all sentences 5 and 7–10 include decimals, as for instance "vingt", "trente", etc.).

The ultimate goal for any speech recognition application will be to accept all correct recognitions and to reject all recognition errors. Unfortunately, this is not feasible in practice since the both groups are overlapping and to distinguish between them is a very difficult trade-off problem causing two main types of application errors: rejecting correct recognitions known as false rejection (FR) errors and accepting recognition errors or so-called false acceptance (FA) errors. Therefore a more modest target will be to find an optimal (minimal error rate) trade-off between FR and FA errors. A rather wide spread approach in speech recognition is to base the acceptance-rejection decision on the comparison of an appropriately defined sentence confidence score (usually in the range 0–100) against an in advance determined so-called rejection threshold (typically 50). If the sentence confidence is above the threshold then it is accepted as correct, otherwise depending on how the application has been designed the user might be asked to repeat or the sentence is just directly rejected as erroneous.

As it can be seen from the appendix, to each word in a given sentence a confidence score has been assigned by the recognition engine. Consequently the sentence confidence scores can be derived from these word confidences via some aggregation process. Table 1 in the appendix contains various confidence scores for each of the 11 sentences corresponding to applying several different aggregation operators. The first aggregation operator used is the arithmetic mean (**M**), which does not exhibit really satisfactory behaviour with an error rate of six FA errors. The choice of another aggregation operator, the geometric mean (**G**), which is known to be less affected by extreme values than the arithmetic mean, leads to a lower number (three) of FA errors, but still too many. The harmonic mean (**H**) corrects for all these FA errors, but unfortunately causes that other three sentences are falsely rejected.

Clearly, none of the above aggregation operators leads to an acceptable trade-off between FR and FA errors. If we assume that the different scores are not equally important than the situation can be modelled by means of weighted aggregation operators. For instance, a well-known phenomenon in speech recognition is that the longer words tend to have more reliable confidences than the ones for rather short words, i.e. the word lengths can serve as weights. However such an approach can cause sometimes a considerable performance degradation due to the fact that wrongly recognized long words with high scores would dominate the final decision. A more robust solution will be to determine the weights via some training process, but as discussed in the introduction, this is not always possible due to the lack of reliable training data. Moreover introducing weights still does not help us in choosing an appropriate aggregation operator since any of the operators mentioned above can be used in a weighted form.

A more pragmatic approach will be to try to combine in some way the three operators **M**, **G** and **H**. For instance, the initial aggregation of the word confidences with these operators will produce three new values. Further these values can be combined again with the same aggregation operators and again until ultimately the difference between the maximum and minimum values will be small enough to stop further aggregation. The confidences for our 11 sentences produced via such an aggregation process are presented in the last column of Table 1. The resulting error rate is very balanced, one FR against one FA.

2.2. Recursive Aggregation Algorithm

The aggregation process described above can be formally defined as follows:

- *Initialization*

  (1) Initial Values: $x_1, \ldots, x_n$, where $x_i \in [0,1]$, for all i.

  (2) Aggregation Operators: $A_1, \ldots, A_m$.

  (3) Initial Aggregation Step:

$$y_1 = A_1(x_1, \ldots, x_n)$$
$$\ldots$$
$$y_m = A_m(x_1, \ldots, x_n).$$

- *Recursion*

  (1) Stop Condition: If $(\max(y_1, \ldots, y_m) - \min(y_1, \ldots, y_m)) < \epsilon$, for a very small $\epsilon \in R^+$, then the final result from the aggregation of $x_1, \ldots, x_n$ is

$$\min(y_1, \ldots, y_m) \approx \max(y_1, \ldots, y_m).$$

  (2) Aggregation Step:

$$\begin{aligned} z_1 &= A_1(y_1, \ldots, y_m) \\ &\ldots \\ z_m &= A_m(y_1, \ldots, y_m). \end{aligned}$$

  (3) Iteration Step:

$$y_1 = z_1, \ldots, y_m = z_m.$$

Now, let us show for which conditions the above algorithm is convergent. It is clear that the convergence is closely related to the ability to compensate between low and high scores. Recall that an aggregation operator $A : R^n \to R$ $(n \geq 2)$ is called *compensatory* if

$$\min(x_1, \ldots, x_n) \leq A(x_1, \ldots, x_n) \leq \max(x_1, \ldots, x_n),$$

for any $(x_1, \ldots, x_n)$ (cf. [12]). Observe that the latter always implies *idempotency*

$$A(x, \ldots, x) = x.$$

The compensatory property alone though, will not be enough to claim that convergence is always guaranteed. It can easily be demonstrated, for instance, that the convergence cannot be achieved whenever min and max operators are used together, although they are both compensatory. Therefore, a further refinement of the compensatory property is required. An aggregation operator $A : R^n \to R$ $(n \geq 2)$ will be referred to as *left-strict compensatory* if

$$\min(x_1, \ldots, x_n) < A(x_1, \ldots, x_n) \leq \max(x_1, \ldots, x_n)$$

and as *right-strict compensatory* if

$$\min(x_1, \ldots, x_n) \leq A(x_1, \ldots, x_n) < \max(x_1, \ldots, x_n),$$

for any $(x_1, \ldots, x_n)$ with at least two different values.
Obviously, an aggregation operator will be *strictly compensatory* whenever it is simultaneously left-strict and right-strict compensatory.

Theorem 1. (Convergence) Any set of continuous and either all left-strict or all right-strict compensatory aggregation operators $A_1, \ldots, A_m$ defines a convergent recursive aggregation process.

Proof. Assume that $A : R^n \to R$ is a MLA operator defined by a set $\{A_1, \ldots, A_m\}$ of continuous and left-strict compensatory operators. Let

$$\alpha_0 = \min(x_1, \ldots, x_n)$$
$$\beta_0 = \max(x_1, \ldots, x_n),$$

and for any further step $i > 0$ of the above algorithm,

$$\alpha_i = \min(y_1, \ldots, y_m)$$
$$\beta_i = \max(y_1, \ldots, y_m).$$

Since the aggregation operators $\{A_1, \ldots, A_m\}$ are all compensatory, we have that at each recursive step $i > 0$ their values will be contained in the interval $[\alpha_i, \beta_i]$. Moreover due to the left-strict compensatory property of the aggregation operators, it holds that

$$\alpha_0 < \alpha_1 < \ldots < \alpha_i < \ldots \quad \text{and} \quad \ldots \leq \beta_i \leq \beta_{i-1} \leq \ldots \leq \beta_0.$$

Therefore we have a $(\beta_i - \alpha_i)$ sequence that is strictly decreasing and non-negative. Further since $\{A_1, \ldots, A_m\}$ are continuous it holds that $\lim_{i \to \infty} (\beta_i - \alpha_i) = 0$. □

Consequently, whenever further in this paper we refer to a *multilayer aggregation operator* (MLA), we will mean an operator defined via the above algorithm and a vector of continuous and either all left-strict or all right-strict compensatory aggregation operators $A_1, A_2, \ldots, A_m$ and it will be denoted as $A_{[A_1, A_2, \ldots, A_m]}$. Note that if all the aggregation operators are left-strict compensatory then the MLA is also left-strict compensatory, and respectively right-strict compensatory aggregation operators determine a right-strict MLA. In general, any MLA is always continuous, compensatory, and consequently idempotent $A_{[A_1, \ldots, A_m]}(x, \ldots, x) = x$. Further MLA is also *idempotent w.r.t. the aggregation operators*, i. e. $A_{[A, \ldots, A]} = A$, for any aggregation operator A.

The result in Theorem 1, though more general, is very close to Matkowski results on iterations of mean-type mapping in [2]. There it has been proven that a sequence of iterates of mean-type mappings defined via continuous and all but (at most) one strict compensatory operators converges to a unique continuous and compensatory mean-type mapping K, which rather obviously satisfies:

$$K(A_1(x_1, \ldots, x_n), \ldots, A_m(x_1, \ldots, x_n)) = K(x_1, \ldots, x_n).$$

Clearly both the results in [2] and Theorem 1 imply that min and max cannot be present at the same time in the definition of MLA otherwise convergence will never be achieved. Moreover, it always holds that

$$A_{[A_1, \ldots, \min, \ldots, A_m]} = \min$$
$$A_{[A_1, \ldots, \max, \ldots, A_m]} = \max,$$

where in the first equation all the aggregation operators are right-strict compensatory and in the second respectively left-strict compensatory. Therefore min and max can be regarded as the extreme values (operators) for MLA.

2.3. Basic Properties

Besides the compensatory and idempotency properties, a number of other useful properties can be associated with an aggregation operator. Consider an aggregation operator $A : R^n \to R$ $(n \geq 2)$.

- *Boundary conditions*:
$$A(0, \ldots, 0) = 0$$
$$A(1, \ldots, 1) = 1.$$

- *Monotonicity* with respect to each argument:
$$x_k < x'_k \Rightarrow A(x_1, \ldots, x_k, \ldots, x_n) \leq A(x_1, \ldots, x'_k, \ldots, x_n).$$
The monotonicity is strict if strict inequality holds.

- *Commutativity*: The indexing of the arguments does not matter, i. e.
$$A(x_1, \ldots, x_n) = A(y_1, \ldots, y_n),$$
for any permutation $(y_1, y_2, \ldots, y_n)$ of $(x_1, x_2, \ldots, x_n)$.

- *Decomposability*:
$$A(x_1, \ldots, x_k, \ldots, x_n) = A(z, \ldots, z, x_{k+1}, \ldots, x_n),$$
where $z = A(x_1, \ldots, x_k)$.

Observe that any idempotent and strictly monotonic aggregation operator is strictly compensatory. Therefore according to Theorem 1 any set of idempotent and strictly monotonic aggregation operators will define an MLA operator. For example, all the mean operators (arithmetic, geometric, quadratic, harmonic, etc.) are idempotent and strictly monotonic and hence strictly compensatory. Besides they are also continuous and commutative.

It is rather straightforward to verify that an MLA operator will satisfy the continuity, the boundary condition, the (strict) monotonicity and commutativity properties if it is composed of aggregation operators that satisfy respectively these properties. Moreover commutative aggregation operators define an MLA operator that is also commutative with respect to the aggregation operators present in its definition. For instance the MLA operator $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ from the appendix can equivalently be denoted as $A_{[\mathbf{M},\mathbf{G},\mathbf{H}]}$ or $A_{[\mathbf{G},\mathbf{H},\mathbf{M}]}$ since **H**, **G**, and **M** are all commutative. In general, any MLA defined via an arbitrary combination of the arithmetic, geometric, quadratic, harmonic, etc. means will satisfy the boundary conditions and will be continuous, strictly compensatory and commutative with respect to both, the input values for aggregation and the aggregation operators present in its definition.

Observe further that the arithmetic, geometric, quadratic, harmonic, etc. means are also decomposable. However, an MLA operator defined via decomposable aggregation operators is not necessarily decomposable. This can easily be verified with a counter example. Let us consider the operator $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ composed by the harmonic, geometric and arithmetic means and a vector $x = (0.8, 0.4, 0.7)$. We have that $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}(0.8, 0.4, 0.7) = 0.606$, while $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}(0.566, 0.566, 0.7) = 0.607$, where $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}(0.8, 0.4) = 0.566$ and therefore $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ is not decomposable.

2.4. Advanced Properties

In this subsection, we consider two interesting monotonicity properties of MLA operators. Thanks to these properties a very useful ranking among MLA operators composed of power means is established in the following section.

Let us now consider a set of aggregation operators $A_1, \ldots, A_k, A'_k \ldots, A_m, A_{m+1}$ which, in accordance to Theorem 1, are assumed to be continuous and either all left-strict or all right-strict compensatory. The first monotonicity property states that when an aggregation operator in the definition of an MLA operator is substituted with a new one that produces higher or lower aggregation values then consequently the MLA operator will produce higher or respectively lower aggregation values. This is formally presented in the proposition below and it can be regarded as monotonicity with respect to each aggregation operator in the MLA definition.

Proposition 1. If $A_1 \ldots, A_k, \ldots, A_m$ are all monotonic aggregation operators and $A_k \leq A'_k$ [2] then it holds

$$A_{[A_1,\ldots,A_k,\ldots,A_m]} \leq A_{[A_1,\ldots,A'_k,\ldots,A_m]}.$$

Proof. Assume that all $A_1, \ldots, A_i, \ldots, A_m$ are monotonic. At the first layer of aggregation, we have that $A_i(x_1, \ldots, x_n) = y_i$, for all i, and $A'_k(x_1, \ldots, x_n) = y'_k$, and moreover $y_k \leq y'_k$. Next due to the monotonicity of A_i we obtain, for any $i \neq k$,

$$z_i = A_i(y_1, \ldots, y_k, \ldots, y_m) \leq A_i(y_1, \ldots, y'_k, \ldots, y_m) = z'_i.$$

Further from $A_k \leq A'_k$ and the monotonicity of A_k

$$\begin{aligned} z_k = A_k(y_1, \ldots, y_k, \ldots, y_m) &\leq A_k(y_1, \ldots, y'_k, \ldots, y_m) \\ &\leq A'_k(y_1, \ldots, y'_k, \ldots, y_m) = z'_k. \end{aligned}$$

Then at the iteration step we will have that $y_i = z_i$ and $y'_i = z'_i$ and $y_i \leq y'_i$, for all i and consequently the same inequalities as above can be applied and so on until the stop condition is reached. □

Now let us consider another interesting type of monotonicity of MLA operators dealing with adding or removing aggregation operators from their definitions. Clearly the latter is closely related to adding and removing arguments from the argument list of an ordinary aggregation operator. In [10], an aggregation operator A is called *monotonic in cardinality* if

$$A(x_1, \ldots, x_n) \leq A(x_1, \ldots, x_n, x_{n+1}).$$

This property says that the addition of elements to the input values cannot result in a decrease of the aggregated value. Similarly, monotonicity in cardinality with respect to the aggregation operators can be introduced, i.e. $A_{[A_1,\ldots,A_m]} \leq A_{[A_1,\ldots,A_m,A_{m+1}]}$.

[2] Further on in this paper, for any two aggregation operators A and B, $A \leq B$ will mean $A(x_1, \ldots, x_n) \leq B(x_1, \ldots, x_n)$, for any $(x_1, \ldots, x_n)$.

Thus adding a new aggregation operator to the MLA operator cannot result in a decrease of the final result.

Although the monotonicity in cardinality property is interesting mathematical abstraction, it is rather strong as a unconditional requirement on an aggregation operator. For instance, no one of the arithmetic, geometric, quadratic, harmonic, etc. means is monotonic in cardinality. Moreover, it is doubtful whether such a property is really desirable in practical applications. In the context of our speech recognition problem, it will imply that longer sentences will in general be assigned higher confidences than any subsentences contained in them. Such a behaviour can only be justified if words with higher confidences have been added. Thus some sort of *conditional monotonicity in cardinality* can be considered, instead.

Lemma 1. If A is an idempotent, decomposable and monotonic aggregation operator then the following properties hold:

$$x_{n+1} \geq A(x_1, \ldots, x_n) \to A(x_1, \ldots, x_n, x_{n+1}) \geq A(x_1, \ldots, x_n)$$
$$x_{n+1} \leq A(x_1, \ldots, x_n) \Rightarrow A(x_1, \ldots, x_n, x_{n+1}) \leq A(x_1, \ldots, x_n).$$

Proof. Let us prove the first implication. Assume $z = A(x_1, \ldots, x_n)$ and hence $x_{n+1} \geq z$. Then applying in turn, the decomposability, monotonicity and idempotency of A, we obtain:

$$\begin{aligned} A(x_1, \ldots, x_n, x_{n+1}) &= A(z, \ldots, z, x_{n+1}) \\ &\geq A(z, \ldots, z, z) = z. \end{aligned}$$

□

The above property claims that adding a new input value higher than the already aggregated value results in a higher final result and analogously, an input value lower than the already obtained one decreases the new aggregation result. This property has also been considered in [11], where it is derived from the monotonicity and self-identity properties. An operator A satisfies the self-identity [11] if for $z = A(x_1, \ldots, x_n)$ it follows that

$$A(x_1, \ldots, x_n, z) = A(x_1, \ldots, x_n) = z.$$

It is easy to see than any idempotent and decomposable aggregation operator satisfies self-identity.

Next, Lemma 1 enables us to derive another interesting monotonicity property of MLA operators. Namely, an increase (respectively decrease) of the MLA scores can be achieved by adding a new aggregation operator that is proven to produce consistently higher (or respectively lower) aggregation values than the aggregation operators already present in the MLA definition.

Proposition 2. If $A_1, \ldots, A_m$ are all decomposable and monotonic aggregation operators then the following properties hold:

$$A_{m+1} \geq \{A_1, \ldots, A_m\} \Rightarrow A_{[A_1, \ldots, A_{m+1}]} \geq A_{[A_1, \ldots, A_m]}$$
$$A_{m+1} \leq \{A_1, \ldots, A_m\} \Rightarrow A_{[A_1, \ldots, A_{m+1}]} \leq A_{[A_1, \ldots, A_m]}.$$

Proof. Let us prove the first property. At the initialization phase, we have $y_i = A_i(x_1, \ldots, x_n)$ and also, due to $A_{m+1} \geq A_i$, $y_{m+1} = A_{m+1}(x_1, \ldots, x_n) \geq y_i$, for $i = 1, \ldots, m$. Then at the first recursive iteration, due to the monotonicity and the idempotence of A_i, it follows that

$$A_i(y_1, \ldots, y_m) \leq A_i(y_{m+1}, \ldots, y_{m+1}) = y_{m+1},$$

for $i = 1, \ldots, m$. Recall that the aggregation operators $A_1, \ldots, A_m, A_{m+1}$ are assumed to be either all left-strict or all right-strict compensatory and hence they are also idempotent. Therefore Lemma 1 can be applied, for $i = 1, \ldots, m$,

$$z_i = A_i(y_1, \ldots, y_m) \leq A_i(y_1, \ldots, y_m, y_{m+1}) = z'_i,$$

and moreover due to $A_{m+1} \geq A_i$,

$$z'_{m+1} = A_{m+1}(y_1, \ldots, y_m, y_{m+1}) \geq A_i(y_1, \ldots, y_m, y_{m+1}) = z'_i \geq z_i.$$

Consequently, at the iteration step, we will have that $y_i = z_i$, for $i = 1, \ldots, m$ and similarly $y'_i = z'_i$, for $i = 1, \ldots, m+1$. Moreover $y'_{m+1} \geq y'_i \geq y_i$, for $i = 1, \ldots, m$. Thus the same approach as at the previous layer can be applied. When the stop condition is met, we will again have $y'_{m+1} \geq y'_i \geq y_i$, for $i = 1, \ldots, m$ and hence $\max(y'_1, \ldots, y'_m, y'_{m+1}) \geq \max(y_1, \ldots, y_m)$. □

As we will see in the following section, from an application point of view this is a rather important result since it provides us with a means to influence the final result by adding or removing aggregation operators from the MLA definition.

3. MULTILAYER AGGREGATION VIA POWER MEANS

As shown in the previous section, any set of idempotent and strictly monotonic aggregation operators define an MLA operator. Moreover MLA operators exhibit special monotonicity properties in case the aggregation operators in their definition are decomposable. Naturally our further investigations are focused on MLA operators defined via a combination of so-called power means operators since these power means are idempotent, continuous, strictly monotonic, commutative and decomposable.

The power means belong to the class of quasi-arithmetic means, see e. g. [1]. If $f : [0,1] \to [-\infty, +\infty]$ is a continuous strictly monotone function, then *quasi-arithmetic means* M_f are defined by

$$M_f(x_1, \ldots, x_n) = f^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} f(x_i)\right).$$

Further, the means M_p, defined by

$$M_p(x_1, \ldots, x_n) = \begin{cases} \left(\frac{1}{n}\sum\limits_{i=1}^{n} x_i{}^p\right)^{1/p} & \text{if } p \neq 0 \\ \left(\prod\limits_{i=1}^{n} x_i\right)^{1/n} & \text{if } p = 0, \end{cases}$$

for $p \in R$ and $x_1, \ldots, x_n \geq 0$, are called *power means.* It can be easily verified that M_1, M_0 and M_{-1} are the *arithmetic* $\mathbf{M} = 1/n \sum_{i=1}^{n} x_i$, the *geometric* $\mathbf{G} = (\prod_{i=1}^{n} x_i)^{1/n}$ and the *harmonic* $\mathbf{H} = n/(\sum_{i=1}^{n} 1/x_i)$ means, respectively. Note that the power means are indeed quasi-arithmetic means with a function f_p, given by:

$$f_p(x) = \begin{cases} x^p & \text{if } p \neq 0 \\ \ln x & \text{if } p = 0. \end{cases}$$

The basic facts about power means are summarized in the following lemma [7].

Lemma 2. The power means M_p have the following properties:

(1) M_p is homogeneous, i. e. $M_p(tx_1, \ldots, tx_n) = tM_p(x_1, \ldots, x_n)$ for all $t > 0$;

(2) If $p \leq q$ then $M_p(x_1, \ldots, x_n) \leq M_q(x_1, \ldots, x_n)$;

(3) For all fixed n and $(x_1, \ldots, x_n)$, the function $p \to M_p(x_1, \ldots, x_n)$ is continuous on $[-\infty, +\infty]$.

Let $A_{[M_{p_1}, \ldots, M_{p_m}]}$ be an MLA operator defined via a combination of power means $M_{p_1}, \ldots, M_{p_m}$. Moreover, without loss of generality, we assume that $p_i \leq p_{i+1}$, for $i = 1, \ldots, m-1$.

Proposition 3. For any vector $(x_1, \ldots, x_n)$ there exists $p \in [p_1, p_m]$, such that

$$A_{[M_{p_1}, \ldots, M_{p_m}]}(x_1, \ldots, x_n) = M_p(x_1, \ldots, x_n).$$

Proof. Recall that the power means are idempotent and strictly monotonic and thus strictly compensatory. Consequently the operator $A_{[M_{p_1}, \ldots, M_{p_m}]}$ will also be strictly compensatory and moreover by Lemma 2(2) we have that

$$M_{p_i}(x_1, \ldots, x_n) \leq M_{p_{i+1}}(x_1, \ldots, x_n),$$

for any $i = 1, \ldots, m-1$. Therefore using the obvious property

$$A_{[M_{p_1}, \ldots, M_{p_m}]}(x_1, \ldots, x_n) = A_{[M_{p_1}, \ldots, M_{p_m}]}(M_1(x_1, \ldots, x_n), \ldots, M_{p_m}(x_1, \ldots, x_n)),$$

it follows that

$$M_{p_1}(x_1, \ldots, x_n) < A_{[M_{p_1}, \ldots, M_{p_m}]}(x_1, \ldots, x_n) < M_{p_m}(x_1, \ldots, x_n).$$

Further due to the continuity property in Lemma 2(3), there exists $p \in [p_1, p_m]$ such that $A_{[M_{p_1}, \ldots, M_{p_m}]}(x_1, \ldots, x_n) = M_p(x_1, \ldots, x_n)$. □

The foregoing proposition highlights an interesting phenomena: an MLA operator, defined via a combination of power means is, in fact, an aggregation operator that compensates between the powers of the different power mean operators in its

definition. This is illustrated in Figure 1, where the function $p \to M_p(c)$ for the confidence vector $c = (94.44, 88.21, 92.86, 98.38, 5.66, 65.76, 98.41, 99.46, 95.21, 72.76)$ of the first sentence in the appendix, is depicted. The aggregation value of 57.98 obtained for this vector via the MLA operator $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ can also be attained via a power mean operator of power $p = -0.34$, resulting between the powers of $\mathbf{H}$ and $\mathbf{G}$.

Further observe that any MLA operator defined via power means can be interpreted as a selftuning family of power mean operators, or a power mean with an variable power, applying different powers to different input vectors. For instance, Table 2 in the appendix contains for each of the 11 sentences from our speech recognition example the sentence confidence scores aggregated with $A_{[\mathbf{H},\mathbf{G}]}$, $A_{[\mathbf{H},\mathbf{M}]}$, $A_{[\mathbf{G},\mathbf{M}]}$, and $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ operators. Consequently, the powers of the corresponding power mean operators generating these confidence scores are presented in Table 3.

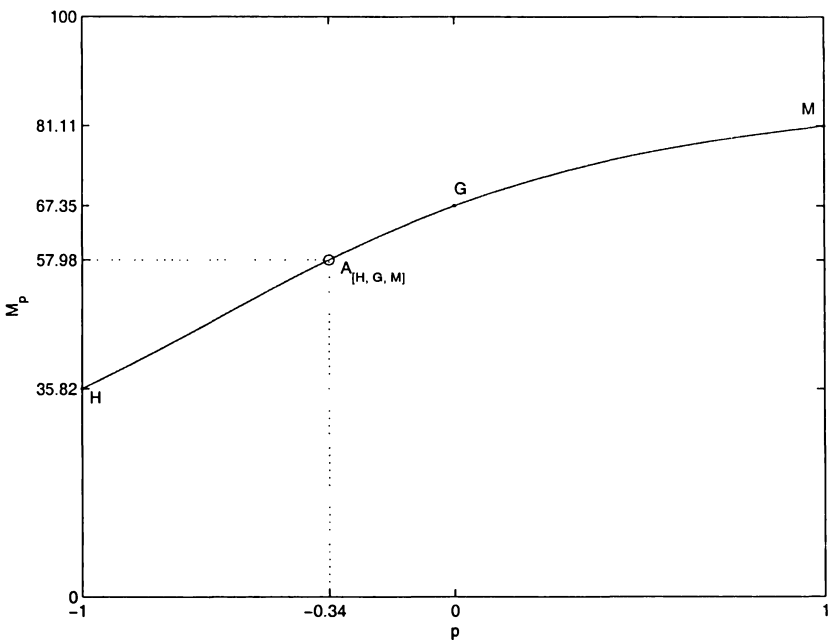

Fig. 1. $p \to M_p(c)$, where
$c = (94.44, 88.21, 92.86, 98.38, 5.66, 65.76, 98.41, 99.46, 95.21, 72.76)$.

It is interesting to observe that the confidence values in the first three columns of Table 2 are ordered in an increasing fashion. This is not a coincidence, due to $\mathbf{H} \leq \mathbf{G} \leq \mathbf{M}$ and Proposition 1 it is always true that

$$A_{[\mathbf{H},\mathbf{G}]} \leq A_{[\mathbf{H},\mathbf{M}]} \leq A_{[\mathbf{G},\mathbf{M}]}.$$

Moreover from Proposition 2, it also holds that (see Table 2)

$$A_{[\mathbf{H},\mathbf{G}]} \leq A_{[\mathbf{H},\mathbf{G},\mathbf{M}]} \leq A_{[\mathbf{G},\mathbf{M}]}.$$

The latter relationship is generalized in the proposition below.

Proposition 4. For any $1 \leq k, l \leq m$, it holds:

$$A_{[M_{p_1},\ldots,M_{p_k}]} \leq A_{[M_{p_1},\ldots,M_{p_m}]} \leq A_{[M_{p_l},\ldots,M_{p_m}]}.$$

Proof. From Lemma 2(2), we have that $M_{p_i} \leq M_{p_{i+1}}$, for any $i = 1,\ldots,m-1$. Then applying in turn the first property in Proposition 2, we obtain the left inequality:

$$A_{[M_{p_1},\ldots,M_{p_m}]} \geq A_{[M_{p_1},\ldots,M_{p_{m-1}}]} \geq \ldots \geq A_{[M_{p_1},\ldots,M_{p_k}]}.$$

The right one can be derived analogously. □

The existing relationships between the MLA operators, composed via all possible combination of **H**, **G** and **M** operators, can be summarized as follows:

$$\mathbf{H} \leq A_{[\mathbf{H},\mathbf{G}]} \leq \{\mathbf{G}, A_{[\mathbf{H},\mathbf{M}]}, A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}\} \leq A_{[\mathbf{G},\mathbf{M}]} \leq \mathbf{M}.$$

The operators in the middle, **G**, $A_{[\mathbf{H},\mathbf{M}]}$ and $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$, appear not to be related in general. For instance, consider two input vectors $x_1 = (0.98, 0.09, 0.08)$ and $x_2 = (0.74, 0.94, 0.06)$. Then, we have that

$$\begin{array}{llll} \mathbf{G}(x_1) & = 0.192 & \mathbf{G}(x_2) & = 0.347 \\ A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}(x_1) & = 0.208 & A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}(x_2) & = 0.315 \\ A_{[\mathbf{H},\mathbf{M}]}(x_1) & = 0.216 & A_{[\mathbf{H},\mathbf{M}]}(x_2) & = 0.302. \end{array}$$

Clearly, **G**, $A_{[\mathbf{H},\mathbf{M}]}$ and $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ values are very close, but there exists no consistent ordering between them. In the example above, $A_{[\mathbf{H},\mathbf{M}]}$ and $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ operators score either above or below **G**. However, a special relationship between the three middle operators can be established in case of two arguments. The latter is generalized in the next proposition.

Proposition 5. For any $(x_1, x_2) \in [0,1]^2$ and $p_i \in R^+$, $i = 1,\ldots,m$, it holds

$$\begin{aligned} M_0(x_1, x_2) &= A_{[M_{-p_m},\ldots,M_{-p_1},M_{p_1},\ldots,M_{p_m}]}(x_1, x_2) \\ &= A_{[M_{-p_m},\ldots,M_{-p_1},M_0,M_{p_1},\ldots,M_{p_m}]}(x_1, x_2). \end{aligned}$$

Proof. Recall that $M_0 = \mathbf{G}$ and consider $p_i \in R^+$, $i = 1,\ldots,m$. The proof is based on the general property $M_{-p_i} \leq \mathbf{G} \leq M_{p_i}$ and the special relationship between these three means in case of two argument list:

$$M_{-p_i}(x_1, x_2) M_{p_i}(x_1, x_2) = (\mathbf{G}(x_1, x_2))^2,$$

Let us now show for instance that

$$M_0(x_1, x_2) = A_{[M_{-p_m}, \ldots, M_{-p_1}, M_0, M_{p_1}, \ldots, M_{p_m}]}(x_1, x_2).$$

Initially we have that

$$\begin{array}{ll} y_1 = M_{-p_m}(x_1, x_2) & y_{m+2} = M_{p_1}(x_1, x_2) \\ \ldots & \ldots \\ y_m = M_{-p_1}(x_1, x_2) & y_{2m+1} = M_{p_m}(x_1, x_2) \end{array}$$

$$y_{m+1} = \mathbf{G}(x_1, x_2) = \sqrt{x_1 x_2},$$

and $y_i y_{2m+2-i} = {y_{m+1}}^2 = x_1 x_2$, for $i = 1, \ldots, m$. Applying the latter at the next aggregation layer, we obtain that

$$\begin{array}{ll} z_1 = M_{-p_m}(y_1, \ldots, y_{2m+1}) & z_{m+2} = M_{p_1}(y_1, \ldots, y_{2m+1}) \\ \ldots & \ldots \\ z_m = M_{-p_1}(y_1, \ldots, y_{2m+1}) & z_{2m+1} = M_{p_m}(y_1, \ldots, y_{2m+1}) \end{array}$$

$$\begin{aligned} z_{m+1} = \mathbf{G}(y_1, \ldots, y_{2m+1}) &= \left(\prod_{i=1}^{2m+1} y_i \right)^{\frac{1}{2m+1}} \\ &= \left(y_{m+1} \prod_{i=1}^{m} y_i y_{2m+2-i} \right)^{\frac{1}{2m+1}} = y_{m+1} = \sqrt{x_1 x_2}. \end{aligned}$$

Similarly, it can be shown that

$$M_{-p_i}(y_1, \ldots, y_{2m+1}) = y_{m+1}^2 \left(\frac{2m+1}{\sum_{j=1}^{2m+1} y_j^{p_i}} \right)^{\frac{1}{p_i}}.$$

Hence it holds that

$$z_i z_{2m+2-i} = M_{-p_i}(y_1, \ldots, y_{2m+1}) M_{p_i}(y_1, \ldots, y_{2m+1}) = {y_{m+1}}^2 = x_1 x_2,$$

for $i = 1, \ldots, m$. Continuing in the same fashion, at each aggregation layer we will have that

$$\mathbf{G}(y_1, \ldots, y_{2m+1}) = \sqrt{x_1 x_2}$$

and therefore the aggregation process will finally converge to $\mathbf{G}(x_1, x_2)$. □

The particular case of the left equality in the above proposition for only two mappings has already been derived in [2], i. e. it has been shown that a sequence of iterates of the mean-type mapping (M_{-p}, M_p) converges to $\mathbf{G}$.

Note that the result in Proposition 5 provides an interesting insight about MLA operators constructed via a set of power mean operators which is symmetrical with respect to G. In general, these MLA operate as a trade-off between the different power mean operators and only in case of two arguments they will converge into $\mathbf{G}$.

Thus, they can be interpreted as some sort of approximation of $\mathbf{G}$. For instance, Figure 2 in the appendix presents the ROC (receiver operator characteristic) curves for each of the operators $\mathbf{G}$, $A_{[\mathbf{H},\mathbf{M}]}$ and $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ in case of a digit string recognition task (the 11 sentences in the appendix are a small subset of the whole experiment of more than 2000 utterances). The ROC curves are constructed by plotting the false rejection rates against the corresponding false acceptance ones for a varying rejection threshold. It can be observed that the rejection performance of $\mathbf{G}$, $A_{[\mathbf{H},\mathbf{M}]}$ and $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ is indeed very comparable, but still not identical as it would be in the two-argument case (for instance sentences consisting of two words only).

4. CONCLUSION

In this paper, we suggest a new way for aggregating a list of values. A vector of aggregation operators is applied repeateadly first over the input values, and then over the results of the previous aggregation step, and so on until certain stop condition is met. We have called this process Multilayer Aggregation (MLA) and determined the conditions under which it is convergent. It has been shown that the properties associated with such MLA operators ultimately depend on the choice of aggregation operators. Interesting monotonicity properties have been derived in case of monotonic, idempotent or/and decomposable aggregation operators. Further MLA operators defined via a combination of power means have been introduced. It has been encountered that these MLA operators compensate between the power values of the different power means and in this way act as a trade-off between the power mean operators present in their definition. Finally, an interesting ranking between some MLA operators defined via power means has been established due to the mentioned above monotonicity properties.

APPENDIX: SPEECH RECOGNITION EXAMPLE

1) *pronounced:* cinq six zéro quatre huit un sept trois deux neuf
recognized: cinq six zéro quatre huit un sept trois deux neuf

correct recognition(CR)

confidences: cinq(94.44) six(88.21) zéro(92.86) quatre(98.38) huit(5.66) un(65.76) sept(98.41) trois(99.46) deux(95.21) neuf(72.76)

2) *pronounced:* cinq huit un deux zéro trois quatre neuf sept six
recognized: cinq huit un deux zéro trois quatre neuf sept six

correct recognition(CR)

confidences: cinq(46.51) huit(27.54) un(68.71) deux(46.88) zéro(47.65) trois(79.53) quatre(86.68) neuf(94.24) sept(96.07) six(84.61)

3) *pronounced:* quatre un cinq six zéro sept trois huit neuf deux
recognized: quatre un cinq six zéro sept trois huit neuf deux

correct recognition(CR)

confidences: quatre(16.25) un(36.44) cinq(16.74) six(46.04) zéro(98.81) sept(75.95) trois(91.85) huit(80.90) neuf(59.03) deux(73.44)

4) *pronounced:* cinq zéro huit six deux trois neuf sept quatre un
recognized: cinq zéro huit six deux trois neuf cinq quatre un huit

recognition error(RE)

confidences: cinq(61.37) zéro(6.82) huit(70.30) six(41.06) deux(52.20) trois(92.94) neuf(93.04) cinq(74.83) quatre(98.50) un(7.02) huit(27.55)

5) *pronounced:* zéro trois quatre vingt quarante neuf trente quatre zéro cinq
recognized: zéro trois quatre cinq zéro neuf trois quatre zéro cinq

recognition error(RE)

confidences: zéro(94.30) trois(99.70) quatre(99.44) cinq(68.40) zéro(5.30) neuf(88.59) trois(98.28) quatre(88.55) zéro(98.12) cinq(43.59)

6) *pronounced:* six cinq neuf deux zéro trois un huit sept quatre
recognized: six cinq neuf neuf zéro trois un huit sept quatre

recognition error(RE)

confidences: six(14.16) cinq(65.98) neuf(95.15) neuf(11.55) zéro(38.55) trois(97.24) un(48.35) huit(23.55) sept(92.81) quatre(94.36)

7) *pronounced:* zéro quatre soixante et onze cinquante sept vingt quatre quatre vingt treize
recognized: zéro quatre cinq un cinq sept cinq quatre quatre quatre

recognition error(RE)

confidences: zéro(54.71) quatre(53.59) cinq(1.80) un(7.89) cinq(13.46) sept(4.44) cinq(74.31) quatre(2.65) quatre(77.66) quatre(5.10)

8) *pronounced:* zéro deux quatre vingt dix sept quatre vingt cinq trente trois soixante deux
recognized: zéro deux quatre six sept quatre cinq trois trois quatre deux

recognition error(RE)

confidences: zéro(99.34) deux(91.02) quatre(19.78) six(79.15) sept(96.00) quatre(72.38) cinq(99.48) trois(29.77) trois(99.65) quatre(0.07) deux(88.06)

9) *pronounced:* zéro trois quarante quatre cinquante deux quatre vingt seize trente trois
recognized: zéro trois quatre quatre cinq deux quatre sept trois

recognition error(RE)

confidences: zéro(98.94) trois(99.71) quatre(12.97) quatre(87.72) cinq(62.06) deux(58.33) quatre(88.85) sept(7.74) trois(96.05)

10) *pronounced:* zéro trois vingt quatre vingt dix neuf cinquante huit vingt quatre
recognized: huit zéro trois cinq quatre huit six neuf quatre huit cinq quatre

recognition error(RE)

confidences: huit(41.70) zéro(99.08) trois(94.36) cinq(93.20) quatre(97.07) huit(49.88) six(86.97) neuf(94.16) quatre(3.29) huit(55.76) cinq(36.48) quatre(57.34)

11) *pronounced:* six cinq neuf deux zéro trois un huit sept quatre
recognized: six cinq neuf deux zéro trois un huit sept quatre

correct recognition(CR)

confidences: six(93.14) cinq(32.85) neuf(88.44) deux(67.46) zéro(99.94) trois(99.44) un(60.66) huit(10.37) sept(59.60) quatre(74.17)

Table 1. Sentence confidence scores and false rejection/acceptance statistics for rejection threshold 50.

Sentence	Classification	M	G	H	$A_{[H,G,M]}$
1)	(CR)	81.11	67.35	**35.82(FR)**	57.98
2)	(CR)	67.84	63.33	58.26	63.02
3)	(CR)	59.55	50.61	**40.40(FR)**	**49.56(FR)**
4)	(RE)	**56.88(FA)**	42.31	24.75	39.04
5)	(RE)	**78.43(FA)**	**63.80(FA)**	33.35	**54.98(FA)**
6)	(RE)	**58.17(FA)**	45.54	32.94	44.35
7)	(RE)	29.56	13.45	6.18	13.67
8)	(RE)	**70.43(FA)**	37.02	0.76	16.95
9)	(RE)	**68.04(FA)**	**51.80(FA)**	30.74	47.65
10)	(RE)	**68.44(FA)**	**53.52(FA)**	25.28	44.94
11)	(CR)	68.61	58.92	**43.37(FR)**	55.96

Table 2. Sentence confidence scores for different MLA operators.

Sentence	$A_{[H,G]}$	$A_{[H,M]}$	$A_{[G,M]}$	$A_{[H,G,M]}$
1)	47.92	53.90	74.07	57.98
2)	60.72	62.87	65.57	63.02
3)	45.07	49.05	54.99	49.56
4)	31.79	37.52	49.32	39.04
5)	44.94	51.14	70.92	54.98
6)	38.48	43.77	51.66	44.35
7)	8.78	13.51	20.72	13.67
8)	2.55	7.32	52.38	16.95
9)	39.23	45.73	59.64	47.65
10)	35.53	41.29	60.28	44.94
11)	50.26	54.55	63.67	55.96

Table 3. Powers of the corresponding power mean operators generating the sentence confidence scores in Table 2.

Sentence	$A_{[H,G]}$	$A_{[H,M]}$	$A_{[G,M]}$	$A_{[H,G,M]}$
1)	−0.64	−0.46	0.35	−0.34
2)	−0.52	−0.09	0.47	−0.06
3)	−0.54	−0.15	0.45	−0.10
4)	−0.56	−0.26	0.42	−0.18
5)	−0.63	−0.44	0.36	−0.32
6)	−0.53	−0.13	0.46	−0.09
7)	−0.48	0.005	0.47	0.016
8)	−0.64	−0.42	0.25	−0.25
9)	−0.57	−0.27	0.40	−0.19
10)	−0.64	−0.46	0.37	−0.33
11)	−0.59	−0.31	0.41	−0.22

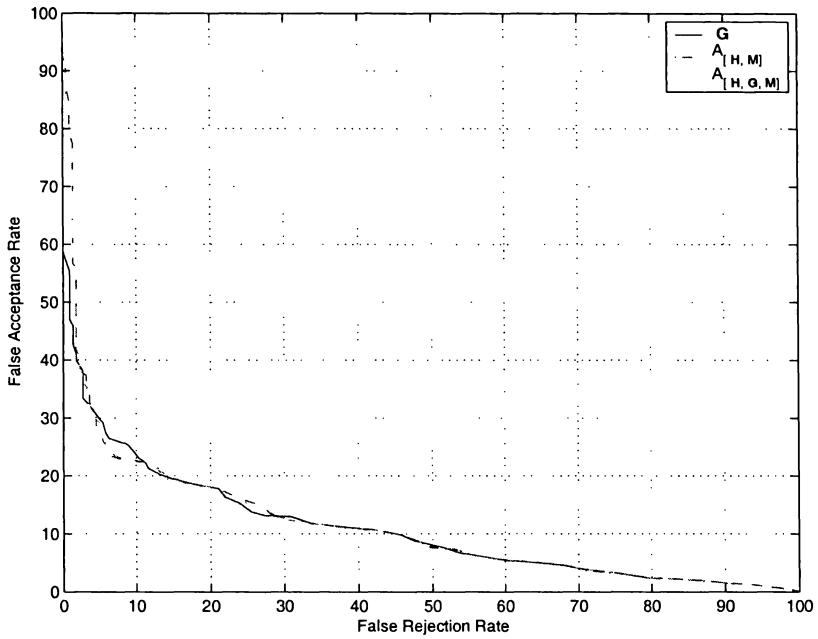

Fig. 2. ROC curves for $\mathbf{G}$, $A_{[\mathbf{H},\mathbf{M}]}$ and $A_{[\mathbf{H},\mathbf{G},\mathbf{M}]}$ for a digit string recognition task.

(Received September 8, 2003.)

REFERENCES

[1] J. C. Fodor and M. Roubens: Fuzzy Preference Modelling and Multicriteria Decision Support. Kluwer, Dordrecht 1994.

[2] J. Matkowski: Iterations of mean-type mappings and invariant means. Ann. Math. Sil. *13* (1999), 211–226.

[3] J. Matkowski: On iteration semi-groups of mean-type mappings and invariant means. Aequationes Math. *64* (2002), 297–303.

[4] R. Mesiar: Aggregation operators: some classes and construction methods. Proceedings of IPMU'2000, Madrid, 2000, pp. 707–711.

[5] M. Mizumoto: Pictorial representations of fuzzy connectives. Part 2: Cases of compensatory and self-dual operators. Fuzzy Sets and Systems *32* (1989), 245–252.

[6] B. Moser, E. Tsiporkova, and K. P. Klement: Convex combinations in terms of triangular norms: A characterization of idempotent, bisymmetrical and self-dual compensatory operators. Fuzzy Sets and Systems *104* (1999), 97–108.

[7] Z. Páles: Nonconvex function and separation by power means. Mathematical Inequalities & Applications *3* (2000), 169–176.

[8] E. Tsiporkova and V. Boeva: Multilayer aggregation operators. In: Proc. Summer School on Aggregation Operators 2003 (AGOP'2003), Alcalá de Henares, Spain, pp. 165–170.

[9] I. B. Turksen: Interval-valued fuzzy sets and 'compensatory AND'. Fuzzy Sets and Systems *51* (1992), 295–307.
[10] R. R. Yager: MAM and MOM opereators for aggregation. Inform. Sci. *69* (1993), 259–273.
[11] R. R. Yager: Noncommutative self-identity aggregation. Fuzzy Sets and Systems *85* (1997), 73–82.
[12] H.-J. Zimmermann and P. Zysno: Latent connectives in human decision making. Fuzzy Sets and Systems *4* (1980), 37–51.

Elena Tsiporkova, DIALOCA SA, Automatic Speech Recognition Research, Pontbeekstraat 4, 1702 Groot-Bijgaarden. Belgium. Current address: *Computational Biology Division, Dept. of Plant Systems Biology, Flanders Institute for Biotechnology, Ghent University, Technologiepark 927, 9052 Ghent. Belgium.*
e-mail: elena.tsiporkova@psb.UGent.be

Veselka Boeva, Department of Computer Systems, Technical University of Plovdiv, Tsanko Dyustabanov 25, 4000 Plovdiv. Bulgaria.
e-mail: veselka_boeva@hotmail.com