Choquet Integrals and OWA Criteria as a Natural (and Optimal) Next Step After Linear Aggregation: A New General Justification

François Modave, Martine Ceberio, and Vladik Kreinovich

Department of Computer Science
University of Texas at El Paso, El Paso, TX 79968, USA,
{fmodave,mceberio,vladik}@utep.edu

Abstract. In areas ranging from multi-criteria decision making to multi-agent decision making, it is necessary to aggregate (combine) utility values corresponding to several criteria and/or several agents. The simplest way to combine these criteria is to use linear aggregation. In many practical situations, linear aggregation does not adequately describe the actual decision making process, so non-linear aggregation is needed.
From the purely mathematical viewpoint, the next natural step after linear functions is the use of quadratic functions, followed by cubic etc. However, in decision making, different non-linearities are usually more adequate, non-linearities like OWA or Choquet integral that use min and max in addition to linear combinations. In this paper, we explain the empirically observed advantage of such functions by proving that under reasonable conditions, such functions are indeed optimal non-linear aggregations.

1 Formulation of the Problem

Traditional approach to decision making: brief reminder. In the traditional approach to decision making, an agent selects an alternative which is the most beneficial to this agent. In order to understand which alternative would be preferred by the agent, it is necessary to describe the agent's preferences in precise terms.

How to describe preferences: general idea. The possibility to describe preferences in precise terms comes from the fact that a decision maker can always decide which of the two alternatives is better (preferable). Thus, if we provide a continuous scale of alternatives, from a very bad to a very good one, then for each alternative in the middle, there should be an alternative on this scale which is, to this decision maker, equivalent to the given one.

How to describe preferences: specific ideas. Such a scale can be easy constructed as follows. We select two alternatives:

– a very negative alternative A_0; e.g., an alternative in which the decision maker loses all his money (and/or loses his health as well), and

- a very positive alternative A_1; e.g., an alternative in which the decision maker wins several million dollars.

Now, for every value $p \in [0,1]$, we can consider a lottery in which we get A_1 with probability p and A_0 with the remaining probability $1-p$. This probability will be denoted by $A(p)$.

For $p = 1$, the probability of the unfavorable outcome A_0 is 0, so the lottery $A(1)$ simply means the very positive alternative A_1. Similarly, for $p = 0$, the probability of the favorable outcome A_1 is 0, so the lottery $A(0)$ simply means the very negative alternative A_0. The larger the probability p, the more preferable the lottery $A(p)$. Thus, the corresponding lotteries $A(p)$ form a continuous 1-D scale ranging from the very negative alternative A_0 to the very positive alternative A_1.

The resulting notion of utility. Practical alternatives are usually better than $A(0) = A_0$ but worse than $A(1) = A_1$: $A(0) < A < A(1)$. Thus, for each practical alternative A, there exists a probability $p \in (0,1)$ for which the lottery $A(p)$ is, to this decision maker, equivalent to A: $A(p) \sim A$. This "equivalent" probability p is called the *utility* of the alternative A and denoted by $u(A)$.

How can we actually find the value of this utility $u(A)$? We cannot just compare A with different lotteries $A(p)$ and wait until we get a lottery for which $A(p) \sim A$: there are many different probability values, so such a comparison would require an impractically long time. However, there is an alternative efficient way of determining $u(A)$ which is based on the following *bisection* procedure.

The main idea of this procedure is to produce narrower and narrower intervals containing the desired value $u(A)$. In the beginning, we only know that $u(A) \in [0,1]$, i.e., we know that $u(A) \in [\underline{u}, \overline{u}]$ with $\underline{u} = 0$ and $\overline{u} = 1$. Let us assume that at some iteration of this procedure, we know that $u(A) \in [\underline{u}, \overline{u}]$, i.e., that $A(\underline{u}) \le A \le A(\overline{u})$. To get a narrower interval, let us take the midpoint $m \stackrel{\text{def}}{=} \dfrac{\underline{u} + \overline{u}}{2}$ of the existing interval and compare $A(m)$ with A.

- If A is better than $A(m)$ ($A(m) \le A$), this means that $m \le u(A)$ and thus, that the utility $u(A)$ belongs to the upper half-interval $[m, \overline{u}]$.
- If A is worse than $A(m)$ ($A \le A(m)$), this means that $u(A) \le m$ and thus, that the utility $u(A)$ belongs to the lower half-interval $[\underline{u}, m]$.

In both cases, we get a new interval containing $u(A)$ whose width is the half of the width of the interval $[\underline{u}, \overline{u}]$. We start with an interval of width 1. Thus, after k iterations, we get an interval $[\underline{u}, \overline{u}]$ of width 2^{-k} that contains $u(A)$. In this case, both endpoints $\underline{u}$ and $\overline{u}$ are 2^{-k}-approximations to $u(A)$. In particular:

- to obtain $u(A)$ with accuracy $1\% = 0.01$, it is sufficient to perform 7 iterations: since $2^{-7} = 1/128 < 0.01$;
- to obtain $u(A)$ with accuracy $0.1\% = 0.001$, it is sufficient to perform 10 iterations: since
$$2^{-10} = 1/1024 < 0.001;$$

- to obtain $u(A)$ with accuracy $10^{-4}\% = 10^{-6}$, it is sufficient to perform 20 iterations: since
$$2^{-20} = 1/(1024)^2 < 10^{-6};$$
- in general, to obtain $u(A)$ with accuracy ε, it is sufficient to perform $\approx -\log_2(\varepsilon)$ iterations.

The numerical value of the utility depends on the choice of extreme alternatives A_0 and A_1. In our definition, the numerical value of the utility depends on the selection of the alternatives A_0 and A_1: e.g., A_0 is the alternative whose utility is 0 and A_1 is the alternative whose utility is 1. What if we use a different set of alternatives, e.g., $A'_0 < A_0$ and $A'_1 > A_1$?

Let A be an arbitrary alternative between A_0 and A_1, and let $u(A)$ be its utility with respect to A_0 and A_1. In other words, we assume that A is equivalent to the lottery in which:

- we have A_1 with probability $u(A)$, and
- we have A_0 with probability $1 - p$.

In the scale defined by the new alternatives A'_0 and A'_1, let $u'(A_0)$, $u'(A_1)$, and $u'(A)$ denote the utilities of A_0, A_1, and A. This means, in particular:

- that A_0 is equivalent to the lottery in which we get A'_1 with probability $u'(A_0)$ and A'_0 with probability $1 - u'(A_0)$; and
- that A_1 is equivalent to the lottery in which we get A'_1 with probability $u'(A_1)$ and A'_0 with probability $1 - u'(A_1)$.

Thus, the alternative A is equivalent to the compound lottery, in which

- first, we select A_1 or A_0 with probabilities $u(A)$ and $1 - u(A)$, and then
- depending on the first selection, we select A'_1 with probability $u'(A_1)$ or $u'(A_0)$ – and A'_0 with the remaining probability.

As the result of this compound lottery, we get either A'_0 or A'_1. The probability p of getting A'_1 in this compound lottery can be computed by using the formula of full probability
$$p = u(A) \cdot u'(A_1) + (1 - u(A)) \cdot u'(A_0) =$$
$$u(A) \cdot (u'(A_1) - u'(A_0)) + u'(A_0).$$
So, the alternative A is equivalent to a lottery in which we get A'_1 with probability p and A'_0 with the remaining probability $1-p$. By definition of utility, this means that the utility $u'(A)$ of the alternative A in the scale defined by A'_0 and A'_1 is equal to this value p:
$$u'(A) = u(A) \cdot (u'(A_1) - u'(A_0)) + u'(A_0).$$
So, changing the scale means a linear re-scaling of the utility values:
$$u(A) \to u'(A) = a \cdot u(A) + b$$

for some $a = u'(A_1) - u'(A_0) > 0$ and $b = u'(A_0)$.

Vice versa, for every $a > 0$ and b, one can find appropriate events A'_0 and A'_1 for which the re-scaling has exactly these values a and b. In other words, utility is defined modulo an arbitrary (increasing) linear transformation.

Utility of an action: a derivation of the expected utility formula. What if an action leads to alternatives $a_1, \ldots, a_m$ with probabilities $p_1, \ldots, p_m$? Suppose that we know the utility $u_i = u(a_i)$ of each of the alternatives $a_1, \ldots, a_m$. By definition of the utility, this means that for each i, the alternative a_i is equivalent to the lottery $A(u_i)$ in which we get A_1 with probability u_i and a_i with probability $1 - u_i$. Thus, the results of the action are equivalent to the "compound lottery" in which, with the probability p_i, we select a lottery $A(u_i)$. In this compound lottery, the results are either A_1 or A_0. The probability p of getting A_1 in this compound lottery can be computed by using the formula for full probability:

$$p = p_1 \cdot u_1 + \ldots + p_m \cdot u_m.$$

Thus, the action is equivalent to a lottery in which we get A_1 with probability p and A_0 with the remaining probability $1 - p$. By definition of utility, this means that the utility u of the action in question is equal to

$$u = p_1 \cdot u_1 + \ldots + p_m \cdot u_m.$$

In statistics, the right-hand of this formula is known as the *expected value*. Thus, we can conclude that the utility of each action with different possible alternatives is equal to the expected value of the utility; see, e.g., [3–5].

Need to combine utilities: case of multi-criterion decision making. In most practical situations, we need to make a decision based on *several* different criteria. For example, a house can be characterized by its location, size, cost, age, etc. Each of these criteria can be described by a numerical characteristic. For each of these characteristic x_i, the above algorithm can describe the value of the utility $u_i(x_i)$ corresponding to this characteristic. Finding a single value of the utility $u_i(x_i)$ with a given accuracy $\varepsilon > 0$ requires (as we have mentioned) a finite number of questions $c \approx -\log_2(\varepsilon)$ to ask the decision maker. Thus, if we consider N different values of this characteristic x_i, we need $c \cdot N$ questions.

These 1-D utility functions enable us only to compare the alternatives characterized by a single characteristic x_i. In practice, we usually a large number M of characteristics $x_1, \ldots, x_M$ describing each alternative. To adequately present the preference between situations described by these multiple characteristics, we must describe N^M different utility values $u(x_1, \ldots, x_M)$ corresponding to N^M possible combinations $(x_1, \ldots, x_M)$. The number of such values grows exponentially with M; so, for large M, it is no longer feasible to ask that many question to the decision maker.

Instead, we can

– ask the decision maker to describe his or her preferences corresponding to each of the M characteristics;

- use the answers to the corresponding questions to form M utility functions $u_1(x_1)$, ..., $u_M(x_M)$ corresponding to M characteristics $x_1, \ldots, x_M$; and then
- combine these N utility functions into a single function describing the decision maker's preference:

$$u(x_1, \ldots, x_M) \approx f(u_1(x_1), \ldots, u_M(x_M))$$

for an appropriate aggregation function $f(u_1, \ldots, u_M)$.

A natural question is: What aggregation function should we choose? This is the question that we will be analyzing and discussing in this paper.

Need to combine utilities: case of multi-agent decision making. Up to now, we have considered situations in which one person makes a decision. In practice, decisions are usually made by a group of people. To find out the corresponding "collective utility" of each alternative, we must therefore ask the group to make selections based on their joint preference. Each group decision requires a lot of discussions, and thus, a large amount of time. When we have a large number of alternatives, the overall time needed to gauge the collective utility values becomes impractically large.

Alternatively, we can try:

- interview each of M decision makers separately, and come up with the corresponding utility functions $u_1(x), \ldots, u_M(x)$; and then
- combine the corresponding utility functions into a single function characterizing the overall utility:

$$u(x) \approx f(u_1(x), \ldots, u_M(x)).$$

Here, the same natural question arises: What aggregation function should we choose?

Simplest case: linear combination. The traditional approach to aggregation is based on the following two facts:

- first, small changes in the values of the characteristics lead to small changes in individual and joint utilities; it is therefore reasonable to assume that the aggregation functions are smooth;
- second, we are mostly interested in comparing similar alternatives – because these are the most difficult to compare; thus, we need to only consider the values of the aggregation function $f(u_1, \ldots, u_M)$ in a small area.

It is well known that on a small domain, each smooth function is well approximated by a linear function – the sum of its constant and linear terms in Taylor expansion. In geometric terms, the smooth graph of the aggregation function can be approximated by its tangent hyperplane – which can be viewed as a graph of some linear function. On the domain of linear size δ, the accuracy of this approximation is determined by the first ignored terms in the Taylor expansion

– quadratic terms $O(\delta^2)$. Thus, the smaller the domain, the more accurate the linear approximation.

In this linear approximation, the aggregation function takes the form

$$f(u_1, \ldots, u_M) = a_0 + a_1 \cdot u_1 + \ldots + a_M \cdot u_M$$

for some coefficients a_0, a_1, $\ldots$, a_M. We have mentioned earlier that utility is defined modulo a starting point, so we can ignore a_0 and conclude that the aggregation is simply a weighted linear combination of component utility values:

$$f(u_1, \ldots, u_M) = a_1 \cdot u_1 + \ldots + a_M \cdot u_M.$$

Need to go beyond linear combination. A linear function is often a good approximation to a general smooth function, but it is still only an approximation. To get a more adequate representation of the decision maker's preferences, we must go beyond linear functions.

A natural question is: how can we expand the class of linear aggregation functions to get a better representation of decision maker's preferences?

Seemingly natural idea: quadratic aggregation functions. Based on the above justification of linear functions – as the sum of 0-th and 1-st order terms in the Taylor expansion – it is reasonable to expect that the reasonable next approximation would include quadratic aggregation functions – followed by cubic etc.

Empirical fact: min**- and** max**-using aggregation functions are more adequate.** It turns out that in decision making, non-quadratic non-linearities are usually more adequate, namely, non-linearities like OWA [7] or Choquet integral [2] that use min and max in addition to linear combinations.

For example, the general OWA combination of two utility values has the form

$$f(u_1, u_2) = w_1 \cdot \min(u_1, u_2) + w_2 \cdot \max(u_1, u_2).$$

Similarly, the general OWA combination of three utility values has the form

$$f(u_1, u_2, u_3) = w_1 \cdot \min(u_1, u_2, u_3) +$$

$$w_2 \cdot \max(\min(u_1, u_2), \min(u_1, u_3), \min(u_2, u_3)) +$$

$$w_3 \cdot \max(u_1, u_2, u_3).$$

What we plan to do. In this paper, we explain the empirically observed advantage of such functions.

2 First Explanation: Why Quadratic Aggregation Functions are Not Very Adequate

Scale invariance: reminder. To understand why quadratic functions are not very adequate, it is sufficient to recall that the utility is defined modulo *two* types of transformations: changing a starting point $u \to u + b$ and changing a scale $u \to \lambda \cdot u$ for some $\lambda > 0$. In the previous section, we only considered the shift $u \to u + b$.

It is also reasonable to consider the re-scaling $u \to u' = \lambda \cdot u$. Namely, the aggregation operation should not depend on which "unit" (i.e., which extreme event A_1) we use to describe utility.

Linear aggregation functions are scale-invariant. The scale-"independence" holds for the above linear combination. Indeed, on the one hand,

– suppose that we start with utility values $u_1, \ldots, u_M$;
– to these values, we apply the above linear aggregation function and get the resulting overall utility $u = a_1 \cdot u_1 + \ldots + a_M \cdot u_M$.

On the other hand,

– we can express the same utility values in a new scale, as
$$u'_1 = \lambda \cdot u_1, \ldots, u'_M = \lambda \cdot u_M;$$
– then, we use the same aggregation function to combine the new utility values; as a result, we get
$$u' = a_1 \cdot u'_1 + \ldots + a_M \cdot u'_M.$$

Substituting the expressions $u'_i = \lambda \cdot u_i$ into this formula, we conclude that
$$u' = a_1 \cdot \lambda \cdot u_1 + \ldots + a_M \cdot \lambda \cdot u_M = \lambda \cdot (a_1 \cdot u_1 + \ldots + a_M \cdot u_M) = \lambda \cdot u.$$

In other words, the new aggregated value u' is exactly the old value u – but described in the new scale.

Scale invariance: a natural requirement on an aggregation function. In view of the above arguments, it is natural to require that in general, the aggregation function should be invariant w.r.t. the same re-scalings. Specifically, we require that the utility
$$u' = f(u'_1, \ldots, u'_M) = f(\lambda \cdot u_1, \ldots, \lambda \cdot u_M)$$
reflect the same degree of preference as the utility $u = f(u_1, \ldots, u_M)$ but in a different scale: $u' = \lambda \cdot u$, i.e.,
$$f(\lambda \cdot u_1, \ldots, \lambda \cdot u_M) = \lambda \cdot f(u_1, \ldots, u_M).$$

Comment. In mathematics, such functions are called *homogeneous* (of first degree).

Why quadratic functions are not very adequate aggregation operators. Now, we are ready to explain why min and max are adequate in describing a standard multi-criterion, while quadratic functions are not:

– min and max are homogeneous functions, while
– a generic quadratic function, e.g., a function $u = u_1^2$ are not homogeneous.

Remaining problem. The scaling idea

– explains why quadratic functions are not very adequate, but
– it does not explain why necessarily aggregation functions consisting of min, max, and linear combinations are empirically the best – there are many different homogeneous functions, why are not these other operations optimal?

3 Which Aggregation Operations Are the Best?

General idea: we need fastest-to-compute aggregation functions. As we have mentioned, one of the reasons why we need aggregation operations in the first place is that a straightforward determination of the multi-dimensional utility function requires too long a time. Therefore, when we select an approximating family for aggregation operations, it is reasonable to select approximating functions which are the fastest to compute.

We need parallelization. One well-known way to speed up computations is to perform them in parallel on several computers. In a parallel computer:

– first, we perform some elementary step(s) in parallel;
– then, we perform some other elementary step(s) in parallel;
– ...
– finally, we perform the last elementary step(s) in parallel.

How to speed up parallel computations. Thus, to speed up computations, we must:

– first, select elementary steps which are the fastest to compute, and
– select a computation steps with the smallest possible number of sequential operations.

Selection of the fastest elementary operations. The fastest computer operations are the ones which are hardware supported, i.e., the ones for which the hardware has been optimized. In modern computers, the hardware supported operations with numbers include elementary arithmetic operations ($+$, $-$, $\cdot$, $/$, etc.), and operations min and max.

In the standard (digital) computer (see, e.g., [1])

– addition of two n-bit numbers requires, in the worst case, $2n$ bit operations: n to add corresponding digits, and n to add carries;
– multiplication, in the worst case, means n additions – by each bit of the second factor; so, we need $O(n^2)$ bit operations;
– division is usually performed by trying several multiplications, so it takes even longer than multiplication;

– finally, min and max can be performed bit-wise and thus, require only n bit operations.

Thus, the fastest elementary operations are addition (or, more generally, linear combination), min, and max.

Thus, the optimal (fastest-to-compute) aggregation functions must be compositions of such fast operations.

Conclusion: we have (almost) justified OWA- and Choquet-type operations. Thus, the optimal aggregation operations are superpositions of linear functions, min, and max – i.e., exactly the operations that are empirically the best.

Remaining questions. Strictly speaking, the above conclusions says that *if* it is possible to approximate an arbitrary homogeneous functions by a composition of such elementary operations, *then* such compositions indeed lead to the fastest-to-compute aggregation functions. Thus, we need to show that such compositions are indeed universal approximators for homogeneous functions.

It is also worth mentioning that the computation time of parallel computations depends not only on the time of each elementary operation, but also on the number of sequential steps (computation layers): the relative advantage of fast elementary operations may be eliminated if we need too many sequential steps. Thus, it is important to find our how many sequential steps we need for the desired approximation.

Let us start answering these questions.

4 Definitions and the Main Result

Definition 1. *A function $f(x_1,\ldots,x_n)$ is called homogeneous if for every x_1, $\ldots$, x_n and for every $\lambda>0$, we have*

$$f(\lambda\cdot x_1,\ldots,\lambda\cdot x_n)=\lambda\cdot f(x_1,\ldots,x_n).$$

Definition 2.

– *By an* elementary operation*, or* a function computable in one step*, we mean one of the following functions:*
  - *a* linear function $f(x_1,\ldots,x_n)=a_1\cdot x_1+\ldots+a_n\cdot x_n$*;*
  - *a* minimum function $f(x_1,\ldots,x_n)=\min(x_{i_1},\ldots,x_{i_m})$*; and*
  - *a* maximum function $f(x_1,\ldots,x_n)=\max(x_{i_1},\ldots,x_{i_m})$.

– *We say that a function $f(x_1,\ldots,x_n)$ is* computable in k steps *if it has a form*

$$f(x_1,\ldots,x_n)=g(h_1(x_1,\ldots,x_n),\ldots,h_m(x_1,\ldots,x_n)),$$

where the function g is computable in one step, and the functions $h_1(x_1,\ldots,x_n)$, $\ldots$, $h_m(x_1,\ldots,x_n)$ are computable in $k-1$ steps.

Comment. By induction over k, one can easily prove that all thus computable functions are homogeneous.

Examples.

– a linear combination is computable in 1 step;
– an OWA combination of two values is computable in 2 steps;
– a general OWA operation is computable in 2 steps.

Definition 3. *Let $k > 0$ be a positive integer. We say that k-layer computations have a* universal approximation property for homogeneous functions *if for every continuous homogeneous function $f(x_1, \ldots, x_n)$, and for every two numbers $\varepsilon > 0$ and $\Delta > 0$, there exists a function $\widetilde{f}(x_1, \ldots, x_n)$ which is computable in k steps and for which $|f(x_1, \ldots, x_n) - \widetilde{f}(x_1, \ldots, x_n)| \le \varepsilon$ for all $x_1, \ldots, x_n$ for which $|x_i| \le \Delta$ for all i.*

Theorem 1. *3-layer computations have a universal approximation property for homogeneous functions.*

Theorem 2.

– *1-layer computations do not have a universal approximation property for homogeneous functions;*
– *2-layer computations do not have a universal approximation property for homogeneous functions.*

Comment. So, by using the above elementary operations, we can compute an arbitrary homogeneous function with a given accuracy in only 3 steps. This is actually exactly as many steps as we need to compute OWA or Choquet aggregations; thus, these empirically successful aggregation operations are indeed as fast as possible – and in this sense optimal.

5 Proof of Theorems 1 and 2

1. Before we start proving, let us notice that the values of the functions

$$\min(x_{i_1}, \ldots, x_{i_m})$$

and $\max(x_{i_1}, \ldots, x_{i_m})$ depend on the order between the values $x_1, \ldots, x_n$. There are $n!$ possible orders, so we can divide the whole n-dimensional space of all possible tuples $(x_1, \ldots, x_n)$ into $n!$ zones corresponding to these different orders.

2. In each zone, an elementary function is linear:

– a linear function is, of course, linear;
– a minimizing function $\min(x_{i_1}, \ldots, x_{i_m})$ is simply equal to the variable x_{i_k} which is the smallest in this zone and is, thus, linear;
– a maximizing function $\max(x_{i_1}, \ldots, x_{i_m})$ is simply equal to the variable x_{i_k} which is the largest in this zone and is, thus, also linear.

3. If a function $f(x_1,\ldots,x_n)$ can be approximated, with arbitrary accuracy, by functions from a certain class, this means that $f(x_1,\ldots,x_n)$ is a limit of functions from this class.

4. Functions computable in 1 step are linear in each zone; thus, their limits are also linear. Since some homogeneous functions are non-linear, we can thus conclude that functions computable in 1 step do not have a universal approximation property for homogeneous functions.

5. Let us now consider functions computable in 2 steps, i.e., functions of the type

$$f(x_1,\ldots,x_n)=g(h_1(x_1,\ldots,x_n),\ldots,h_m(x_1,\ldots,x_n)),$$

where g and h_i are elementary functions.

Since there are three types of elementary functions, we have three options:

- $g(x_1,\ldots,x_m)$ is a linear function;
- $g(x_1,\ldots,x_m)$ is a minimizing function; and
- $g(x_1,\ldots,x_m)$ is a maximizing function.

Let us consider these three options one by one.

5.1. Let us start with the first option, when $g(x_1,\ldots,x_m)$ is a linear function. Since on each zone, each elementary function h_i is also linear, the composition $f(x_1,\ldots,x_n)$ is linear on each zone.

5.2. If $g(x_1,\ldots,x_m)$ is a minimizing function, then on each zone, each h_i is linear and thus, the composition $f(x_1,\ldots,x_n)$ is a minimum of linear functions. It is known that minima of linear functions are concave; see, e.g., [6]. So, within this option, the function $f(x_1,\ldots,x_n)$ is concave.

5.3. If $g(x_1,\ldots,x_m)$ is a maximizing function, then on each zone, each h_i is linear and thus, the composition $f(x_1,\ldots,x_n)$ is a maximum of linear functions. It is known that maxima of linear functions are convex; see, e.g., [6]. So, within this option, the function $f(x_1,\ldots,x_n)$ is convex.

6. In each zone, functions computable in 2 steps are linear, concave, or convex. The class of all functions approximable by such 2-step computable functions is the class of limits (closure) of the union of the corresponding three classes: of linear, concave, and convex sets. It is known that the closure of the finite union is the union of the corresponding closures. A limit of linear functions is always linear, a limit of concave functions is concave, and a limit of convex functions is convex. Thus, by using 2-step computable functions, we can only approximate linear, concave, or convex functions. Since there exist homogeneous functions which are neither linear nor concave or convex, we can thus conclude that 2-step computable functions are not universal approximators for homogeneous functions.

7. To complete the proof, we must show that 3-steps computable functions are universal approximators for homogeneous functions. There are two ways to prove it.

7.1. First, we can use the known facts about concave and convex functions [6]:

- that every continuous function on a bounded area can be represented as as a difference between two convex functions, and
- that every convex function can be represented as a maximum of linear functions – namely, all the linear functions which are smaller than this function.

This facts are true for general (not necessarily homogeneous) functions. For homogeneous functions $f(x_1, \ldots, x_n)$, one can easily modify the existing proofs and show:

- that every homogeneous continuous function on a bounded area can be represented as as a difference between two convex homogeneous functions, and
- that every homogeneous convex function can be represented as a maximum of homogeneous linear functions – namely, all the homogeneous linear functions which are smaller than this function.

Thus, we can represent the desired function $f(x_1, \ldots, x_n)$ as the difference between two convex homogeneous functions $f(x_1, \ldots, x_n) = f_1(x_1, \ldots, x_n) - f_2(x_1, \ldots, x_n)$. Each of these convex functions can be approximated by maxima of linear functions and thus, by functions computable in 2 steps. Substraction $f_1 - f_2$ adds the third step, so $f(x_1, \ldots, x_n)$ can indeed be approximated by functions computable in 3 steps.

To prove that a function $f(x_1, \ldots, x_n)$ can be represented as a different between two convex functions, we can, e.g., first approximate it by a homogeneous function which is smooth on a unit sphere

$$\{(x_1, \ldots, x_n) : x_1^2 + \ldots + x_n^2 = 1\},$$

and then take $f_1(x_1, \ldots, x_n) = k \cdot \sqrt{x_1^2 + \ldots + x_n^2}$ for a large k. For smooth functions, convexity means that the Hessian matrix – consisting of its second derivatives $\dfrac{\partial^2 f}{\partial x_i \partial x_j}$ – is positive definite.

For sufficiently large k, the difference

$$f_2(x_1, \ldots, x_n) = f_1(x_1, \ldots, x_n) - f(x_1, \ldots, x_n)$$

is also convex – since its second derivatives matrix is dominated by positive definite terms coming from f_1. Thus, the difference $f_1 - f_2 = f$ is indeed the desired difference.

7.2. Another, more constructive proof, is, for some $\delta' > 0$, to select a finite δ'-dense set of points $e = (e_1, \ldots, e_n)$ on a unit square. For each such point, we build a 2-step computable function which coincides with f on the corresponding ray $\{\lambda \cdot (e_1, \ldots, e_n) : \lambda > 0\}$. This function can be obtained, e.g., as a minimum of several linear functions which have the right value on this ray but change drastically immediately outside this ray.

For example, let $f_0(x)$ be an arbitrary homogeneous linear function which coincides with $f(x)$ at the point e – and thus, on the whole ray. To construct the corresponding linear functions, we can expand the vector e to an orthonormal basis e, e', e'', etc., and take linear functions $f_0(x) + k \cdot (e' \cdot x)$ and $f_0(x) - k \cdot (e' \cdot x)$

for all such e' (and for a large $k > 0$). Then, the minimum of all these functions is very small outside the ray.

We then take the maximum of all these minima – a 3-step computable function.

The function $f(x_1, \ldots, x_n)$ is continuous on a unit sphere and thus, uniformly continuous on it, i.e., for every $\varepsilon > 0$, there is a δ such that δ-close value on the unit sphere lead to ε-close values of f. By selecting appropriate δ' and k (depending on δ), we can show that the resulting maximum is indeed ε-close to f.

The theorem is proven.

Acknowledgments

This work was supported in part by NSF grant HRD-0734825, by Grant 1 T36 GM078000-01 from the National Institutes of Health, by the Japan Advanced Institute of Science and Technology (JAIST) International Joint Research Grant 2006-08, and by the Max Planck Institut für Mathematik.

References

1. Cormen, T.H., Leiserson, C. E., Rivest, R. L., Stein, C.: *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2001.
2. Grabisch, M., Murofushi, T., Sugeno, M. eds.: *Fuzzy Measures and Integrals*, Physica-Verlag, Berlin-Heidelberg, 2000.
3. Keeney, R. L., Raiffa, H.: *Decisions with Multiple Objectives*, John Wiley and Sons, New York, 1976.
4. Luce, R. D., Raiffa, H.: *Games and Decisions: Introduction and Critical Survey*, Dover, New York, 1989.
5. Raiffa, H.: *Decision Analysis*, Addison-Wesley, Reading, Massachusetts, 1970.
6. Rockafeller, R. T.: *Convex Analysis*, Princeton University Press, Princeton, New Jersey, 1970.
7. Yager, R. R., Kacprzyk, J., eds.: *The Ordered Weighted Averaging Operators: Theory and Applications*, Kluwer, Norwell, Massachusetts, 1997,