Main Ideas Behind OWA Lead to a Universal and Optimal Approximation Scheme

Ronald R. Yager[1] and Vladik Kreinovich[2]
[1]Machine Intelligence Institute
Iona College, New Rochelle, NY 10801-1890
[2]Department of Computer Science, University of Texas at El Paso
El Paso, TX 79968, contact email vladik@cs.utep.edu

Abstract

Ordered Weighted Averaging (OWA) operators have been successfully applied in many practical problems. We explain this empirical success by showing that these operators are indeed guaranteed to work (i.e., are universal), and that these operators are the best to use (in some reasonable sense).

1 Aggregation is needed

In many areas of science and engineering, we have several estimates $x_1, \ldots, x_n$ for the same quantity x. These estimates may come from measurements and/or they may come from experts. and we want to combine them into a single (better) estimate y.

Several techniques have been successfully used to aggregate different estimates. The most widely used idea is to take an arithmetic average

$$y = \frac{x_1 + \ldots + x_n}{n}.$$

In the arithmetic average, we combine all the estimates with equal weights. In some practical situations, it makes sense to give move weight to consistent estimates and less weight to estimates that are far away from the consensus of the majority. For example, in some sports competitions, the lowest and the highest scores are deleted, and the average of the remaining values is taken as the resulting aggregate. In more precise terms, this aggregating operation $y = f(x_1, \ldots, x_n)$ can be described as

$$f(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \frac{x_{(2)} + x_{(3)} + \ldots + x_{(n-1)}}{n-2},$$

where $x_{(1)}$ is the smallest of the n values $x_1, \ldots, x_n$, $x_{(2)}$ is the second smallest, etc.

Instead of simply ignoring the outstanding estimates (i.e., assigning them 0 weight), we can give them smaller weight depending on their deviation from the others. For example, we can compute the mean $\bar{x}$ and the standard deviation σ of the original n estimates, and then combine then with weights proportional to $s(|x_i - \bar{x}|/\sigma)$, where $s(z)$ is a decreasing function.

How can we describe different possible aggregation techniques?

2 Linearization: what is it and why it is a widely used application tool

One of the main tools of applied mathematics is linearization; see, e.g., [1]. The need for some tool of this type comes from the fact that the actual dependence $y = f(x_1, \ldots, x_n)$ between physical quantities can be very complex and thus, very difficult to analyze. However, it is usually smooth (differentiable). As a result, when we know the approximate values $\widetilde{x}_1, \ldots, \widetilde{x}_n$ of the quantities $x_1, \ldots, x_n$, then we can expand the dependence of y on x_i into Taylor series in x_i:

$$f(x_1, \ldots, x_n) = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) +$$
$$(x_1 - \widetilde{x}_1) \cdot f_{,1} + \ldots + (x_n - \widetilde{x}_n) \cdot f_{,n} +$$
$$\frac{1}{2} \cdot (x_1 - \widetilde{x}_1)^2 \cdot f_{,11} + \ldots,$$

where $f_{,1}$ means a partial derivative over x_1, $f_{,11}$ means second partial derivative, etc. Since x_i is close to $\widetilde{x}_i$, the differences $x_i - \widetilde{x}_i$ are small, hence we can safely ignore terms which are quadratic (or of higher

order) in terms of these differences. As a result, we conclude that with a reasonable accuracy, the original complex dependence can be represented by a linear function:

$$f(x_1,\ldots,x_n) \approx f(\widetilde{x}_1,\ldots,\widetilde{x}_n)+$$
$$(x_1-\widetilde{x}_1)\cdot f_{,1}+\ldots+(x_n-\widetilde{x}_n)\cdot f_{,n}.$$

For aggregation operations, we thus get a justification for linear aggregation functions like arithmetic average.

One practically useful feature of linearization is that it is *consistent* in the following sense. Instead of directly aggregating n estimates $x_1,\ldots,x_n$, we can divide them into groups (not necessarily disjoint), aggregate the values within each group, and then aggregate the results. This is a natural thing to do if we have a large number of different experts: we can first get an average of experts from the same area, and then try to reconcile the resulting averages.

From the mathematical viewpoint, instead of applying a single aggregation operation to n estimates $x_1,\ldots,x_n$ to get y, we first apply k different aggregation operations to this data to get k intermediate aggregation results $y_1,\ldots,y_k$, and then use another aggregation operation to combine these k intermediate results into a single estimate y. If all these $k+1$ aggregation operations are linear, then the resulting aggregation operation $x_1\ldots,x_n\to y$ is also linear: indeed, if a quantity y linearly depends on $y_1,\ldots,y_k$, and each of the quantities $y_1,\ldots,y_k$ linearly depends on the quantities $x_1,\ldots,x_n$, then the dependence of y on $x_1,\ldots,x_n$ is also linear. In mathematical terms, this consistency can be expressed as follows: a composition of linear functions is linear.

It is worth mentioning that vice versa, every linear function can be represented as a composition of two basic operations: addition and multiplication by a constant.

3 Linearization by itself does not work well for fuzzy logic operations: the appearance of OWA operations

In fuzzy logic, linearization by itself does not work well. Indeed, the simplest aggregation operations of fuzzy logic – min and max – are not smooth and therefore, cannot be well approximated by linear functions.

How can we combine the convenience of linear approximations with the necessity to also have min and max? A natural idea is to add min an max to the list of basic operations. In other words, instead of linear functions, i.e., functions obtained by composition of addition and multiplication by a number, we consider functions which are compositions of addition, multiplication by a constant, min, and max.

Each such function is piece-wise linear. We will therefore call such functions *p-linear* (p is short of "piecewise"). An important particular class of such operations are Ordered Weighted Averaging (OWA) operations (see, e.g., [4, 5]), i.e., operations of the type

$$f(x_1,\ldots,x_n)=$$
$$a_0+a_1\cdot x_{(1)}+a_2\cdot x_{(2)}+\ldots+a_n\cdot x_{(n)}, \quad (1)$$

where $a_0,a_1,\ldots,a_n$ are constants.

The values $x_{(1)},\ldots,x_{(n)}$ can be easily described in terms of min and max:

- $x_{(1)}=\min(x_1,\ldots,x_n)$;
- $x_{(2)}=\max(x(1),\ldots,x(n))$,
  where $x(i)$ is the minimum of all the values except i-th, i.e.:
  - $x(1)\stackrel{\text{def}}{=}\min(x_2,x_3,\ldots,x_n)$,
  - $x(2)\stackrel{\text{def}}{=}\min(x_1,x_3,x_4,\ldots,x_n)$,
  - $\ldots$,
  - $x(n)\stackrel{\text{def}}{=}\min(x_1,\ldots,x_{n-1})$;
- $x_{(3)}=\max(x(1,2),x(1,3),\ldots,x(n-1,n))$,
  where $x(i,j)$ is the minimum of all the values except i-th and j-th;
- etc.

Examples: $\min(x_1,x_2)$ can be described as an OWA operation corresponding to $a_0=0$, $a_1=1$, and $a_2=0$; $\max(x_1,x_2)$ can be described as $a_0=a_1=0$, $a_2=1$; arithmetic average corresponds to $a_0=0$, $a_1=\ldots=a_n=1/n$, and the above sports average corresponds to $a_0=a_1=a_n=0$ and $a_2=\ldots=a_{n-1}=1/(n-2)$.

Usually, only averaging OWA-operations are considered, i.e., operations for which $a_0=0$, values

$a_1, \ldots, a_n$ are non-negative, and $a_1 + \ldots + a_n = 1$. However, a similar approach can be used to describe other aggregation operations, not only averaging ones. In this case, it makes sense to consider the most general operations of type (1), without imposing any restrictions on the real numbers a_i.

4 In contrast to linearization, OWA description is not fully consistent

Linear aggregation is consistent in the sense that for linear aggregation operations, composition is always a linear function. For OWA operations, this is not always true: if the dependence of y on $y_1, \ldots, y_k$ is described by formulas of the type (1), and the dependence of each y_i on $x_1, \ldots, x_n$ is also described by a similar formula, then the dependence of z on $x_1, \ldots, x_n$ may be more complex than the formula (1).

So, we must consider not only the original OWA operations, but also compositions of different OWA operations. One can easily prove that the class of such compositions coincides with the class of all compositions of addition, multiplication by a constant, min, and max – i.e., with the class of all possible p-linear functions.

5 Natural questions: Are these compositions universal? optimal in any reasonable sense?

Not every aggregation operation is an OWA operation. For example, the weighted average with the weight $s(|x - \bar{x}|/\sigma)$ is not an OWA operation. We know one way of designing non-OWA operations: composition of several OWA ones. The first natural question is therefore: are such compositions universal approximations? I.e., can we approximate an arbitrary continuous function by compositions of this type? In this paper, we give a positive answer to this question.

The proof of this result is similar to the proofs of universal approximation results for fuzzy systems (see, e.g., [2] and references therein).

The next question is: how good is this approximation? It is known that fuzzy systems are universal approximators, that neural networks are universal approximators, that polynomials are universal approximators, etc. Is there any advantage in using OWA operators?

The empirical fact is that OWA operators are indeed useful, but it would be nice to have a theoretical explanation for that. We prove that, in some reasonable sense, OWA-type operations are indeed optimal – namely, they are (in some reasonable sense) the fastest to compute.

The proof of this result is similar to the proof explaining why fuzzy control is sometimes useful even when there is no expert knowledge, as a good approximation tool; see, e.g., [3].

6 It is often important to aggregate fast; how can we do it?

In many real-life situations – e.g., in automated control – we must make urgent decisions based on the values of certain critical physical characteristics. To make an informed decision, we often make several measurements of the same characteristic. Thus, in order to make a decision, we must aggregate these measurement results into a single value. Since a decision needs to be made urgently, we must aggregate fast.

A natural way to increase the speed of the computations is to perform computations *in parallel* on several processors. To make the computations really fast, we must divide the algorithm into parallelizable steps, each of which requires a small amount of time.

What are these steps? Inside the computer, each computation is represented as a sequence of hardware implemented operations: arithmetic operations $a + b$, $a - b$, $a \cdot b$, a/b, and $\min(a, b)$ and $\max(a, b)$. The time required for each operation, crudely speaking, corresponds to the number of bit operations that have to be performed:

- Operations min and max are the fastest. Indeed, min and max of two $n-$bit binary numbers can be done in n binary operations: we compare the bits from the highest to the lowest, and as soon as they differ, the number that has 0 as opposed to 1 is the desired minimum: e.g., the minimum of 0.10101 and 0.10011 is 0.10011, because in the third bit, this number has 0 as opposed to 1. Similarly, max is an $n-$bit operation.

- Operations $-$ and $+$ are second fastest. To add two $n-$bit binary numbers, we need n bit additions, and also potentially, n bit additions for carries. Totally, we need about $2n$ bit operations.

- Multiplication by a constant can be implemented as a sequence of additions, so it is also a fast operation.

- Multiplication of two general n-bit numbers is implemented as a sequence of n additions of n-bit numbers (again, basically in the same manner as we do it manually). It requires n^2 bit operations and is thus much slower than $+$.

- Division is done by successive multiplication, comparison and subtraction (basically, in the same way as we do it manually), so, it is an even slower operation than multiplication.

The fastest possible aggregation operations are, therefore, operations that only use min and max. Alas, not all aggregation operations can be thus represented, because if we start with n numbers $x_1, \ldots, x_n$ and apply only min and max, we end up with one of these n numbers, so even an arithmetic average will not be covered.

Thus, to describe generic aggregation operations, we must use not only the fastest computer operations (min and max), but also the second fastest: $+$, $-$, and multiplication by a constant. In other words, we must consider p-linear functions.

Which p-linear functions are the fastest to compute? On each time step of a parallel computer, processor perform several different hardware supported operations. The most time-consuming operations are $+$, $-$, and multiplication by a constant. Therefore, to speed up computations, we must consider computations in which a linear combination is computed on the smallest possible number of time steps. We will show that it is sufficient to have only one such time step: exactly as many as for the original OWA operations.

In other words, we will show that it is sufficient to first compute min and max, the compute (in parallel) several linear combinations, and then again apply min and max. Let us describe our result in exact terms.

7 Definition and the main result

Definition. *By a fast function, we mean a composition of* min *and* max. *We say that a function $f(x)$, $x \in R^n$, is computable with a single non-fast time step if this function can be represented as*

$$f(x) = F(g_1(x), \ldots, g_N(x)),$$

where $F(y_1, \ldots, y_N)$ is a fast function, and each function g_k is a linear combination of fast functions, i.e., has the form

$$g_k(x) = a_{k0} + a_{k1} \cdot h_{k1}(x) + \ldots + h_{km_k}(x)$$

for some fast functions $h_{ki}(x)$.

In other words, first we compute fast functions h_{ij}, then we compute (in parallel) all linear combinations to compute g_k, and then we apply fast operations to combine g_k into the desired value f. In particular, if we take $N = 1$, $h_{1i}(x_1, \ldots, x_n) = x_{(i)}$, and $F(x) = x$, we get an arbitrary OWA operation.

The following result shows that these fast-to-compute operations can indeed approximate an arbitrary continuous operation with an arbitrary accuracy:

Theorem. *For every real number ε, for every box $B = [a_1, b_1] \times \ldots \times [a_n, b_n]$, and for every continuous function $f : B \to R$, there exists a function $\widetilde{f}(x)$ that is $\varepsilon-$close to $f(x)$ on B and that is computable with a single non-fast time step.*

In other words, the fastest-to-compute non-trivial functions – i.e., functions computable with a single non-fast time step – are universal approximators.

8 Proof

The main construction behind the proof is as follows: we pick a small value $\alpha > 0$. Then, for each of n coordinates x_i, we select a grid of values $a_i + (1/2) \cdot \alpha$, $a_i + (3/2) \cdot \alpha$, $a_i + (5/2) \cdot \alpha$, $\ldots$, until we reach b_i. This selection is equivalent to dividing the interval $[a_i, b_i]$ into subintervals

$$[0, \alpha], [\alpha, 2\alpha], \ldots, [k \cdot \alpha, (k+1) \cdot \alpha], \ldots$$

of length α, and selecting midpoints of these subintervals. Thus, for each coordinate, we select $N_i = \lceil (b_i - a_i)/\alpha \rceil$ different values.

We then take all possible points with these coordinates, i.e., we take $N = N_1 \cdot N_2 \cdot \ldots \cdot N_n$ points from the box B. We will denote these grid points by $x^{(1)}, \ldots, x^{(N)}$. This selection is equivalent to subdividing the box B into N small boxes $b_k = b_{k1} \times \ldots \times b_{kn}$ of size $\alpha \times \ldots \times \alpha$, and selecting a midpoint of each small box as $x^{(k)}$.

To design the approximating function, we need another parameter – a large real number $M > 0$. Once this number is selected, as the desired approximating function, we take the function

$$\widetilde{f}(x) \stackrel{\text{def}}{=} \max(g_1(x), \ldots, g_N(x)), \tag{2}$$

where, for each k,

$$g_k(x) \stackrel{\text{def}}{=}$$

$$\min(g_{k0}(x), g^-_{k1}(x), g^+_{k1}(x), \ldots, g^-_{kn}(x), g^+_{kn}(x)), \tag{3}$$

$$g_{k0}(x) \stackrel{\text{def}}{=} f(x^{(k)}); \tag{4}$$

$$g^-_{ki}(x) \stackrel{\text{def}}{=} f(x^{(k)}) + M \cdot (x_i - (x_i^{(k)} - \alpha/2)); \tag{5}$$

$$g^+_{ki}(x) \stackrel{\text{def}}{=} f(x^{(k)}) - M \cdot (x_i - (x_i^{(k)} + \alpha/2)). \tag{6}$$

Here:

- each variable x_i is, of course, a (trivial case of) a fast function;
- the functions $g_{k0}(x)$, $g^-_{ki}(x)$, and $g^+_{ki}(x)$ are linear functions, i.e., linear combinations of fast functions x_i;
- finally, the transitions – from $g_{k0}(x)$, $g^-_{ki}(x)$, and $g^+_{ki}(x)$ to $g_k(x)$, and then from $g_k(x)$ to $\widetilde{f}(x)$ – are performed by using min and max, i.e., are fast.

Thus, the above formulas describe how we can compute the function $\widetilde{f}(x)$ with a single non-fast time step.

To complete the proof, we must therefore find α and M for which this fast-to-compute function $\widetilde{f}(x)$ is ε-close to the given continuous function $f(x)$.

Since the function f is continuous, there exists a $\delta > 0$ such that if $|x_i - x'_i| \le \delta$ for all i, then

$$|f(x_1, \ldots, x_n) - f(x'_1, \ldots, x'_n)| \le \varepsilon. \tag{8}$$

Let us take $\alpha = \delta/2$. As M, we will then take

$$M = \frac{1}{\alpha} \cdot \max_{i \neq j} |f(x^{(i)}) - f(x^{(j)})|. \tag{9}$$

Let us show that for this choice, for every point $x \in B$, we have

$$|\widetilde{f}(x) - f(x)| \le \varepsilon. \tag{10}$$

Let x be an arbitrary point from the box B. Small boxes b_k cover the entire box B, thus, the point x belongs to one of these small boxes. Let b_k be the corresponding subbox, and let $x^{(k)}$ be its midpoint. Then, the box b_k has the form

$$\left[x_1^{(k)} - \frac{\alpha}{2}, x_1^{(k)} + \frac{\alpha}{2}\right] \times \ldots \times \left[x_n^{(k)} - \frac{\alpha}{2}, x_n^{(k)} + \frac{\alpha}{2}\right].$$

Since $x \in b_k$, we conclude that for every i, we have $x_i \in [x_i^{(k)} - \alpha/2, x_i^{(k)} + \alpha/2]$, i.e., $x_i^{(k)} - \alpha/2 \le x_i \le x_i^{(k)} + \alpha/2$.

For such values, $x_i - (x_i^{(k)} - \alpha/2) \ge 0$, hence, by definition of the function $g^-_{ki}(x)$, we have $g^-_{ki}(x) \ge f(x^{(k)}$, i.e., (due to definition of $g_{k0}(x)$), $g^-_{ki}(x) \ge g_{k0}(x)$. Similarly, we have $g^+_{ki}(x) \ge g_{k0}(x)$, and so, due to (3), $g_k(x) = g_{k0}(x) = f(x^{(k)})$. Thus, due to (2), $\widetilde{f}(x) \ge g_k(x) = f(x^{(k)})$.

Since $|x_i - x_i^{(k)}| \le \alpha/2$ and $\alpha = \delta/2$, we conclude that $|x_i - x_i^{(k)}| < \delta$, and hence, due to our choice of δ, that

$$|f(x) - f(x^{(k)})| \le \varepsilon, \tag{11}$$

in particular, that $f(x^{(k)}) \ge f(x) - \varepsilon$. From $\widetilde{f}(x) \ge f(x^{(k)})$, we can now conclude that $\widetilde{f}(x) \ge f(x) - \varepsilon$.

To complete the proof, we must show that $\widetilde{f}(x) \le f(x) + \varepsilon$. Since $\widetilde{f}(x)$ is defined as the maximum of N values $g_l(x)$, it is sufficient to prove that

$$g_l(x) \le f(x) + \varepsilon \tag{12}$$

for each of N functions $g_1(x), \ldots, g_N(x)$. We will prove this by considering two possible cases:

- the first case is when for every i, we have $|x_i - x_i^{(l)}| \le (3/2) \cdot \alpha$;
- the second case is when for some i, we have $|x_i - x_i^{(l)}| > (3/2) \cdot \alpha$.

Let us first consider the first case, when for every i, we have $|x_i - x_i^{(l)}| \le (3/2) \cdot \alpha$. Due to our choice

of α, we have $|x_i - x_i^{(l)}| \leq \delta$, hence, due to our choice of δ, we have $|f(x) - f(x^{(l)})| \leq \varepsilon$. Therefore, $f(x^{(l)}) \leq f(x) + \varepsilon$. By definition, the function $g_{l0}(x)$ is a constant function equal to $f(x^{(l)})$, hence the last inequality can be rewritten as $g_{l0}(x) \leq f(x) + \varepsilon$. Due to (3), the value $g_l(x)$ is the smallest of several values including $g_{l0}(x)$. Since $g_{l0}(x) \leq f(x) + \varepsilon$, we can thus conclude that $g_l(x) \leq f(x) + \varepsilon$. So, the inequality (12) is proven for the first case.

Let us now consider the second case, when for some i, we have $|x_i - x_i^{(l)}| > (3/2) \cdot \alpha$. In this case, there are two possible subcases:

- either $x_i - x_i^{(l)} > (3/2) \cdot \alpha$,
- or $x_i - x_i^{(l)} < -(3/2) \cdot \alpha$.

In the first subcase, $x_i - (x_i^{(l)} + \alpha/2) > \alpha$, hence, due to (6), we get

$$g_{li}^{+}(x) = f(x^{(l)}) - M \cdot (x_i - (x_i^{(l)} + \alpha/2)) <$$

$$f(x^{(l)}) - M \cdot \alpha. \qquad (13)$$

Due to our choice of M (formula (9)), we have $M \geq (1/\alpha) \cdot |f(x^{(l)} - f(x^{(k)}|$, hence

$$M \geq \frac{1}{\alpha} \cdot f(x^{(l)} - f(x^{(k)}),$$

and (13) implies that

$$g_{li}^{+} < f(x^{(l)}) - (f(x^{(l)}) - f(x^{(k)})) = f(x^{(k)}).$$

Thus, the value $g_l(x)$ which is defined (by formula (3)) as the smallest of several values including $g_{li}^{-}(x)$, is also smaller than $f(x^{(k)})$: $g_l(x) < f(x^{(k)})$. From (11), we conclude that $f(x^{(k)}) \leq f(x) + \varepsilon$ and hence, we get the desired inequality $g_l(x) \leq f(x) + \varepsilon$.

In the second subcase, this equality similarly follows from considering a function $g_{li}^{-}(x)$. In all cases, the inequality (12) is proven, so the theorem is proven as well.

9 Conclusion

OWA operators have been successfully applied in many practical problems. We explain this empirical success by showing two things:

- that these operators are indeed guaranteed to work (i.e., are universal), and
- that these operators are the best to use (in some reasonable sense).

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209 and grant NCC2-1232, by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants numbers F49620-95-1-0518 and F49620-00-1-0365, and by NSF grants CDA-9522207, ERA-0112968 and 9710940 Mexico/Conacyt.

The authors are very thankful to the anonymous referees for their useful comments.

References

[1] R. P. Feynman, R. B. Leighton, and M. L. Sands, *The Feynman Lectures On Physics*, Addison-Wesley, Redwood City, CA, 1989.

[2] V. Kreinovich, G. C. Mouzouris, and H. T. Nguyen, "Fuzzy rule based modeling as a universal approximation tool", In: H. T. Nguyen and M. Sugeno (eds.), *Fuzzy Systems: Modeling and Control*, Kluwer, Boston, MA, 1998, pp. 135–195.

[3] R. N. Lea and V. Kreinovich, "Intelligent Control Makes Sense Even Without Expert Knowledge: an Explanation", *Reliable Computing*, 1995, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), pp. 140–145.

[4] H. T. Nguyen and V. Kreinovich, "Kolmogorov's Theorem and its impact on soft computing", In: R. R. Yager and J. Kacprzyk, *The Ordered Weighted Averaging Operators: Theory and Applications*, Kluwer, Boston, MA, 1997, pp. 3–17.

[5] R. R. Yager and J. Kacprzyk, *The Ordered Weighted Averaging Operators: Theory and Applications*, Kluwer, Boston, MA, 1997.