Expansible and Reductible Computable Aggregation Rules

Ramón González del Campo, Luis Garmendia, and Javier Montero

Universidad Complutense de Madrid, Spain
rgonzale@estad.ucm.es,
lgarmend@fdi.ucm.es,
monty@mat.ucm.es

Abstract. The aggregation operator have been considered from a computable point of view. The important condition that the computation is friendly when portions of data are inserted o deleted to the list of values to aggregate is considered.

1 Introduction

The families aggregation operators [2] have been widely studied from a mathematical point of view. The definition of an aggregation operator is a mapping:

$$Ag : [0,1]^n \to [0,1]$$

monotone non-decreasing and with boundary conditions $Ag(0,\ldots,0) = 0$ and $Ag(1,\ldots,1) = 1$.

There are many families of aggregation functions, as means, OWAs [11], Choquet Integral, triangular norms and conforms, uninorms and nullnorms.

Some authors [6, 9, 10] propose to revisit the aggregation definition because it is defined to aggregate a fixed number of values, and some computable aggregation functions should be studied to be useful when that fixed number is not constant, but variable, introducing the term aggregation rule. The aggregation have been also seen from a formal specification and implementation method [8] and classified into basic and non basic, and also by the functional or imperative paradigms of programming.

The main purpose of this paper is to study families of aggregation that can be defined by functions or algorithms for any number of aggregated values, and can reuse the computations when some values are added or deleted. In big data environments those aggregations can also be computed successfully by parts in distributed parallel in many nodes (mapping) and reduced easily in a multi node Apache Hadoop system using the map-reduce programming paradigm.

If aggregation operators that usual databases compute using SQL in relational databases, such MAX, MIN, AVG, SUM must be maintained when the data is charged, it is worth it to study whether those operations should be computed from the beginning or reusing the already computed values.

The recursive aggregations rules [3] [4] [5] [7] give several recursive operators that reuses the already computed value when one value or row is added to the database, now we define the Extensible and Reductible Computable aggregations rules that can be used when many values or rows are inserted or deleted.

2 Preliminaries

The concept of computational complexity cost of an algorithm is reminded from definitions 1 to 5.

Definition 1 *Let $f : \mathbb{N} \to \mathbb{R}$ be a function. The set of functions called to be in the order of f, $O(f)$, are defined as follows:*

$$O(f) \equiv \{g : \mathbb{N} \to \mathbb{R} \mid \exists c \in \mathbb{R}, n_0 \in \mathbb{N}\ \forall n \geq n_0\ g(n) \leq cf(n)\}$$

The order of f contains all the functions that grows up slower than f.

Definition 2 *Let $f : \mathbb{N} \to \mathbb{R}$ be a function. The computational complexity of f, $\Theta(f)$, is defined as follows:*

$$\Theta(f) \equiv \{g : \mathbb{N} \to \mathbb{R} \mid \exists c, d \in \mathbb{R}, n_0 \in \mathbb{N}\ \forall n \geq n_0\ df(n) \leq g(n) \leq cf(n)\}$$

Definition 3 *Let $f, g : \mathbb{N} \to \mathbb{R}$ be two functions. It is said f has a complexity lower than g if $O(f) \subset O(g)$*

Corollary 1 *If $f_1(n) = 1$, $f_2(n) = log(n)$ and $f_3(n) = n$ then:*

$$O(f_1) \subset O(f_2) \subset O(f_3)$$

Definition 4 *Let $g : \mathbb{N} \to \mathbb{R}$ be a function. Constant complexity, linear complexity and logarithmic complexity are defined as follows:*

- g has constant complexity if g belongs to $\Theta(1)$, i.e, if g grows up as fast as $f(n) = 1$.
- g has linear complexity if g belongs to $\Theta(n)$, i.e, if g grows up as fast as $f(n) = n$.
- g has logarithmic complexity if g belongs to $\Theta(log(n))$, i.e, if g grows up as fast as $f(n) = log(n)$.

Definition 5 *The computational complexity cost of an algorithm is the order of the function that gives the computing time of the algorithm.*

Definition 6 *Let T be a t-norm. The residuated implication $\widehat{T} : [0,1]^2 \to [0,1]$ is defined as follows:*

$$\widehat{T}(x, y) = \sup\{z : T(x, z) \leq y\}$$

Definition 7 *[1] A list is an Abstract Data Type that represents a sequence of values. A list can be defined by its behavior and its implementation must provide the following operations:*

- *Test whether a list is empty or not.*
- *Add a value.*
- *Remove a value.*
- *Compute the length (number of values) of a list.*

Definition 8 *[3] A left-recursive connective rule is a family of connective operators:*

$$(Ag : [0,1]^n \to [0,1])_{n>1}$$

such that there exists a sequence of binary operators:

$$(L_n : [0,1]^2 \to [0,1])_{n>1}$$

verifying:

- $Ag(a_1, a_2) = L_2(a_{\pi(1)}, a_{\pi(2)})$
- $Ag(a_1 \ldots, a_n) = L_n(Ag(a_{\pi(1)}, \ldots, a_{\pi(n-1)}), a_{\pi(n)})$ *for all* $n > 2$

for some ordering rule π.

In similar way a right-recursive connective rule can be defined.

A right-recursive connective and left-recursive connective rule is called recursive connective rule.

3 Computable Aggregation Rules

As an alternative definition of aggregation operator, a computable aggregation concept is defined.

Definition 9 *A computable aggregation rule is a mapping* $Ag : L \to [0,1]$ *such that* $Ag(L) = F(L)$ *where* F *is an algorithm that verify the aggregation properties for each list of values* $L = \{x_1, \ldots, x_n\}$.

The computable aggregation rules can be classified by the computational complexity of the algorithm (linear, logarithmic, parabolic...) or by it programming paradigm (iterative, recursive, rules, map-reduce...), or even by algorithm strategy (divide and conquest, backtraking, greedu...).

In real life the data analysis is done in datawarehouses or Big Data Apache Hadoop systems where new data can be loaded monthly, dayly or continuously from operational databases or commercial ETL products.

4 Recursive Computable Aggregation Rules

When computing a recursive connective rule it is much more easy to use the same operator L_i every time that we can denote $\phi : [0,1]^2 \to [0,1]$. If a new value to aggregate is added it can considered a recursive computation:

- $Ag(a_1, a_2) = \phi(a_1, a_2)$

– $Ag(a_1, ..., a_n) = \phi(Ag(a_1, ..., a_{n-1})), a_n)$ for $n > 2$.

So it is easier to define it from just a binary operator ϕ rather than a family of connective operators L_i.

An alternative definition of recursive connective rule is the following.

Definition 10 *Let $L = \{x_1, \ldots, x_n\}$ be a list of values. A computable aggregation rule Ag is recursive if there exists a mapping $\phi : [0,1]^2 \to [0,1]$ such that:*

$$Ag(L) = \begin{cases} x_1, & \text{if } lenght(L) = 1; \\ \phi(Ag(x_1, ..., x_{n-1})), x_n), & \text{if } lenght(L) > 1. \end{cases}$$

Remark. Note that if $lenght(L) = 2$ then $\phi(Ag(x_1), x_2) = \phi(x_1, x_2)$.

Example 1 *Example of implementation using a static list:*

```
float phi(float x, float y);
float aggregation(float L[], int n){
    if (lenght(L)==1){
        return L[0];
    else
        return phi(aggregation(L,n-1),L[n]);
}
```

Remark. Note that a computable aggregation rule is a recursive computable aggregation rule if its algorithm F is recursive.

5 Expansible Computable Aggregation Rules

The expansible computable aggregations rules can be useful to maintain aggregations operations calculations when sets of new data is added to a decision support system.

Definition 11 *A computable aggregation rule is expansible if there exists a mapping $\phi : [0,1]^2 \to [0,1]$ satisfying the following property for all list L_1 and L_2:*

$$Ag(L_1 \cup L_2) = \phi(Ag(L_1), Ag(L_2))$$

where ϕ is an algorithm with linear or lower computational complexity cost.

Lemma 1 *If an aggregation rule is expansible computable then it is a recursive computable aggregation rule.*

Proof. Trivial, it is just the particular case where $length(L_2) = 1$ using the same ϕ operator.

Proposition 1 *The arithmetic mean is a expansible computable aggregation rule with $\phi(x, y) = \frac{x*length(L_1)+y*length(L_2)}{length(L_1)+length(L_2)}$.*

Proof. Let L_1 and L_2 two lists such that $L_1 = \{l_1 + l_2 + \ldots + l_n\}$ and $L_2 = \{l_{n+1} + l_2 + \ldots + l_m\}$.

Then:

- $Ag(L_1) = \frac{1}{length(L_1)}(l_1 + l_2 + \ldots + l_n)$ so: $(l_1 + l_2 + \ldots + l_n) = Ag(L_1) * length(L_1)$
- $Ag(L_2) = \frac{1}{length(L_2)}(l_{n+1} + \ldots + l_m)$ so: $(l + l_{n+1} + \ldots + l_m) = Ag(L_2) * length(L_2)$

Then:

$Ag(L_1 \cup L_2) =$

$= \frac{1}{length(L_1)+length(L_2)}(l_1 + l_2 + \ldots + l_n + l_{n+1} + \ldots + l_m)$

$= \frac{1}{lengtht(L_1)+length(L_2)}(Ag(L_1) * length(L_1) + Ag(L_2) * length(L_2))$

So: $\phi(x, y) = \frac{x*length(L_1)+y*length(L_2)}{length(L_1)+length(L_2)}$

Proposition 2 *The geometric mean is a computable aggregation rule with* $\phi(x, y) = (x^{length(L_1)} * y^{length(L_2)})^{\frac{1}{length(L_1)+length(L_2)}}$.

Proof. Let L_1 and L_2 two lists such that $L_1 = \{l_1 + l_2 + \ldots + l_n\}$ and $L_2 = \{l_{n+1} + l_2 + \ldots + l_m\}$.

Then:

- $Ag(L_1) = \sqrt[length(L_1)]{l_1 * \ldots l_n} = (l_1 * \ldots * l_n)^{\frac{1}{length(L_1)}}$ so: $l_1 * \ldots * l_n = Ag(L_1)^{length(L_1)}$
- $Ag(L_2) = \sqrt[length(L_2)]{l_{n+1} + l_2 + \ldots + l_m} = (l_{n+1} + l_2 + \ldots + l_m)^{\frac{1}{length(L_2)}}$ so: $l_{n+1} * \ldots * l_m = Ag(L_2)^{length(L_2)}$

Then:

$Ag(L_1 \cup L_2) =$

$= (l_1 * l_2 * \ldots * l_n * l_{n*1} * l_2 * \ldots * l_m)^{\frac{1}{length(L_1)+length(L_2)}}$

$= ((Ag(L_1))^{length(L_1)} * ((Ag(L_2))^{length(L_2)})^{\frac{1}{length(L_1)+length(L_2)}}$

So: $\phi(x, y) = (x^{length(L_1)} * y^{length(L_2)})^{\frac{1}{length(L_1)+length(L_2)}}$

Proposition 3 *The OWA aggregation operators are not expansible aggregation rules.*

Proof. The OWA aggregation operators are not expansible aggregation rules, as a sort of values taking is required a $O(n * log(n))$ complexity cost, so it is not linear cost.

Proposition 4 *Aggregation rules defined by the associative property of binary operations, such as triangular norms* T*, where* $Ag(x_1, \ldots x_n) = T(x_n, \ldots T(x_3, T(x_2, x_1)))$ *and triangular conorms* S *are expansible aggregation rules and* $\phi = T$.

Proof. By the associative property, values can be computed in any order, or by subsets.

6 Reductible Computable Aggregation Rules

In some real life datawarehouses some historical data is deleted. For example, banks usually store the accounts movements for two years for legal requirements, and at the end of the month data of the last month is loaded, and the last month two years ago is deleted.

Definition 12 *Let L_2 be a sublist of L_1. A reducible aggregation rule is a computable aggregation satisfying the following property:*

$$Ag(L_1 \setminus L_2) = \psi(Ag(L_1), Ag(L_2))$$

where ψ is an algorithm with linear complexity cost or lower.

Proposition 5 *The arithmetic mean is a reductible computable aggregation rule with $\psi(x, y) = \frac{x*length(L_1)-y*length(L_2)}{length(L_1)-length(L_2)}$.*

Proof. Trivial

Proposition 6 *The product is a reductible computable aggregation rule with $\widehat{prod}(y, x)$ where $\widehat{prod}$ is the residuated operator of the product t-norm.*

Proof. Trivial

Example 2 *Let $L_1 = \{0.6, 0.5, 0.8\}$, $L_2 = \{0.8\}$, $x = Prod(L_1) = 0.24$ and $y = Prod(L_2) = 0.8$*

Then $Prod(L_1 \setminus L_2) = Prod(\{0.6, 0.5\}) = \widehat{Prod}(y, x) = 0.3$.

Proposition 7 *The minimum t-norm is not a reductible aggregation rule.*

Proof. Let $L_1 = \{x_1, x_2\}$ and $L_2 = \{x_2\}$.

If $x_2 < x_1$, then $\min(L_1) = \min(L_2) = x_2$ and so $\min(L_1 \setminus L_2) = x_1$ and it can not be computed from $\min(L_1)$ and $\min(L_2)$

Remark. The reason that the information of value x_1 is lost in the aggregation of L_1.

Proposition 8 *The Lukasiewicz t-norm is not a reductible aggregation rule.*

Proof. Let $L_1 = \{0.6, 0.5, 0.8\}$, $L_2 = \{0.8\}$. Then $W(L_1) = 0$ and $W(L_2) = 0.8$ from those values it can not be computed $W(L_1 \setminus L_2) = W(0.6, 0.5) = 0.1$.

Remark. The reason is that $W(L_1) = 0$, so some aggregated values information is lost.

7 Conclusions

Some aggregation rules are studied from a computational point of view, specially its friendly behavior when the amount of data is growing or decreasing, which is called expansible and reducible.

Some aggregations are not friendly in this situations, as OWAs or Choquet integral, as a sort of all the values must be performed having a non linear complexity cost $\Theta(n * log(n))$. Many aggregation rules are shown to be expansible as usual means, triangular norms and conorms, but only a few of them are reducible.

Acknowledgment

We thank the support of the research project TIN2012-32482 and LODISCO TIN2014-56381-REDT.

References

1. G. Barnett and L. Del Tonga. *Data Structures and Algorithms*. DotNetSlackers, 2008.
2. G. Beliakov, A. Pradera, and T. Calvo. *Aggregations Functions: A guide for Practitioners*. Springer, 2007.
3. V. Cutello and J. Montero. Recursive connective rules. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, (14):3–20, 1999.
4. A. del Amo, J. Montero, and E. Molina. Representation of consistent recursive rules. *European Journal of Operational Research*, 130(1):29–53, 2001.
5. D. Gómez and J. Montero. A discussion on aggregations operators. *Kybernetika*, 40:107–120, 2004.
6. D. Gómez, J. Montero, J. Tinguaro, and K. Rojas. Stability in aggregation operators. In *Advances in Computational Intelligence - 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2012, Catania, Italy, July 9-13, 2012, Proceedings, Part III*, pages 317–325, 2012.
7. A. Kolesárová, R. Mesiar, and J. Montero. Sequential aggregation of bags. *Information Sciences*, 294:305–314, 2015.
8. V. López, J. Montero, and J. Tinguaro. Formal specification and implementation of computational aggregation functions. In *International FLINS Conference on Decision Making and Soft Computing*, 2010.
9. K. Rojas, D. Gómez, J. Montero, and J. Tinguaro. Strictly stable families of aggregation operators. *Fuzzy Sets and Systems*, 228:44–63, 2013.
10. J. Tinguaro, V. López, D. Gómez, B. Vitoriano, and J. Montero. A computational definition of aggregation rules. In *FUZZ-IEEE 2010, IEEE International Conference on Fuzzy Systems, Barcelona, Spain, 18-23 July, 2010, Proceedings*, pages 1–5, 2010.
11. R.R. Yager. Families of owa operators, fuzzy sets and systems. *Fuzzy Sets and Systems*, (59):125–148, 1993.