Using linear programming for weights identification of generalized Bonferroni means in R

Gleb Beliakov and Simon James

School of Information Technology, Deakin University
221 Burwood Hwy, Burwood 3125, Australia
{gleb,sjames}@deakin.edu.au

Abstract. The generalized Bonferroni mean is able to capture some interaction effects between variables and model mandatory requirements. We present a number of weights identification algorithms we have developed in the R programming language in order to model data using the generalized Bonferroni mean subject to various preferences. We then compare its accuracy when fitting to the journal ranks dataset.

Keywords: Aggregation functions, means, generalized Bonferroni mean, weights identification, least absolute deviation (LAD) fitting.

1 Introduction

In decision-making and information processing contexts, the need often arises to merge multiple inputs into a single representative output. For more sophisticated aggregation functions to find use in everyday applications, ways of interpreting their behavior and implementation tools need to be developed to make them accessible to practitioners. In recent years, such developments include the *Kappalab* R package by Grabisch et al. [13], and *AOTool* and *fmtools* by Beliakov [1]. These tools allow the parameters and weights of aggregation functions to be automatically learned from data and used to predict unknown values or analyze the datasets.

The Bonferroni mean [11] is an aggregation function with the ability to model mandatory requirements, i.e. we can ensure that some criteria are at least partially satisfied for a high overall score. Since it was generalized by Yager in [21] a number of publications have followed, with generalizations refined in [8, 15, 22], extensions to higher level fuzzy sets in [7, 18–20] and lattices [6]. As well as

modeling mandatory requirements, the Bonferroni mean could also be useful as a non-linear function which is able to capture interaction effects. Indeed, in its original form the terms of the function are similar to those in statistics used to model interaction between pairs of variables in regression models.

There are a number of ways to construct aggregation functions for applications. Sometimes the parameters can be specified by experts while in other cases we may have an existing dataset and we want a model that reflects the relationship between the inputs and outputs. In the latter case, we can use optimization techniques and perform fitting in order to learn the parameters or weights of the function we wish to use in the model. Due to the composition of the generalized Bonferroni mean, however, a number of issues arise in attempting to learn its weights from data. In general, the problem is not one that can be framed as a linear or quadratic program. In this paper we give an overview of some approaches we have taken and implemented in the R programming language [16]. The fitting algorithms we have developed, as well as our datasets and preprocessing techniques are available as R source files at our website[1].

We investigate three simplifications that allow the problem to be formulated as a linear programming problem and compare the accuracy of the resulting Bonferroni means to other functions with a real data set.

The paper will be structured as follows. In Section 2, we give an overview of aggregation functions and how they can be fit to data. It is here that we also provide the definition of the generalized Bonferroni mean. In Section 3 we show how the weights identification problem for the Bonferroni mean can be formulated linearly, using the same techniques as are employed in fmtool. We then show how the Bonferroni mean compares to other functions when fit to some journal rankings datasets in Section 4. As well as fitting to each full journal set, we also use 10-fold cross-validation tests to show the robustness of the fitting process. We give a brief summary in Section 5.

[1] http://aggregationfunctions.wordpress.com .

2 Preliminaries

We will give an overview here of the definitions and methods that will be used throughout the rest of the paper. In particular, we are concerned with Aggregation functions and techniques for learning their parameters from data.

2.1 Aggregation Functions

The study of aggregation functions for decision making and information processing applications has become increasingly widespread. A number of recent monographs give an overview of their use and properties, [9,14,17]. We will consider aggregation functions defined over the unit interval.

Definition 1. *An aggregation function* $f : [0,1]^n \rightarrow [0,1]$ *is a function non-decreasing in each argument and satisfying* $f(0,\ldots,0) = 0$ *and* $f(1,\ldots,1) = 1$.

Definition 2. *An aggregation function is considered to be:* averaging *where* $\min(\mathbf{x}) \leq f(\mathbf{x}) \leq \max(\mathbf{x})$, conjunctive *where* $f(\mathbf{x}) \leq \min(\mathbf{x})$, disjunctive *where* $f(\mathbf{x}) \geq \max(\mathbf{x})$, *and* mixed *otherwise.*

Due to the monotonicity of aggregation functions, averaging behavior is equivalent to idempotency, i.e. $f(t,t,...,t) = t$.

In this paper, we are primarily concerned with averaging aggregation functions, although the Bonferroni mean uses the product $f(x,y) = xy$ in its composition which is one of the archetypical conjunctive functions.

Well known means include the arithmetic mean (also commonly referred to as the average) and the median. The arithmetic mean, as well as geometric means and power means can be expressed as special cases of the weighted quasi-arithmetic mean. We provide its definition here.

Definition 3. *For a strictly monotone continuous generating function* $\phi : [0,1] \rightarrow [-\infty,\infty]$ *and weighting vector* $\mathbf{w}$*, the weighted quasi-arithmetic mean is given by,*

$$QAM_{\mathbf{w}}(\mathbf{x}) = \phi^{-1}\left(\sum_{i=1}^{n} w_i \phi(x_i)\right). \tag{1}$$

Special cases include weighted arithmetic means, where $\phi(t) = t$, weighted power means where $\phi(t) = t^p$ and weighted geometric means (i.e. $G(\mathbf{x}) = \prod_{i=1}^{n} x_i^{w_i}$) if $\phi(t) = -\ln t$. The weights w_i are usually non-negative and sum to one.

On the other hand, the median, maximum and minimum operators can be expressed as special cases of the ordered weighted averaging (OWA) operator. Rather than weight arguments according to their position or source, the OWA allocates a weight depending on the relative size of the input. It was formally defined by Yager in 1988 [23].

Definition 4. *For a weighting vector* $\mathbf{w}$*, the ordered weighted averaging (OWA) operator is given by,*

$$OWA_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_{(i)}, \tag{2}$$

where the parentheses $(.)$ *indicate a reordering of the inputs such that* $x_{(1)} \geq x_{(2)} \geq \ldots \geq x_{(n)}$*.*

Special cases include the maximum when $\mathbf{w} = (1, 0, \ldots, 0)$, the minimum when $\mathbf{w} = (0, \ldots, 0, 1)$ and the median if $w_i = 1$ for $i = \frac{n+1}{2}$ and 0 otherwise where n is odd, and $w_i = 0.5$ for $i = \frac{n}{2}, \frac{n}{2} + 1$ and 0 otherwise where n is even.

The Bonferroni mean was defined in 1950 [11] and later generalized by Yager and others in the computational intelligence and decision making field. In its original form, it is defined as follows.

Definition 5. *Let* $p, q \geq 0$ *and* $x_i \geq 0, i = 1, \ldots, n$*. The Bonferroni mean is the function*

$$B^{p,q}(\mathbf{x}) = \left(\frac{1}{n(n-1)} \sum_{i,j=1, i \neq j}^{n} x_i^p x_j^q \right)^{\frac{1}{p+q}}. \tag{3}$$

In the case of $p = q$ for $n = 2$ the Bonferroni mean is equivalent to the geometric mean. For $q = 0$ (or $p = 0$), it will reduce to a power mean and can therefore express functions such as the arithmetic mean ($p = 1$), quadratic mean ($p = 2$) and the limiting case of the geometric mean $p = 0$. As the ratio $\frac{p}{q}$ approaches infinity (or 0), the

mean approaches the maximum operator. When $n > 2$, there must exist at least one pair (i, j) such that $x_i, x_j > 0$, for the Bonferroni mean to return a non-zero output $B^{p,q}(\mathbf{x}) > 0$. It is this property that makes it possible for the generalizations of the Bonferroni mean to express mandatory requirements.

In [8], the Bonferroni mean was expressed as a composed aggregation function, generalizing it in terms of two means and a conjunctive function. With this construction, the function is able to model partial conjunction [12] with respect to any number of arguments, i.e. we can specify mandatory requirements that must at least partially be fulfilled for the function to have a non-zero output.

The notation $\mathbf{x}_{j\neq i}$ is used to denote the vector in $[0,1]^{n-1}$ that includes the arguments from $\mathbf{x} \in [0,1]^n$ in each dimension except the i-th, $\mathbf{x}_{j\neq i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$.

Definition 6. *[8]. Let $\mathbb{M}$ denote a 3-tuple of aggregation functions $< M_1, M_2, C >$, with $M_1 : [0,1]^n \to [0,1]$, $M_2 : [0,1]^{n-1} \to [0,1]$ and $C : [0,1]^2 \to [0,1]$, with the diagonal of C denoted by $C_*(t) = C(t,t)$ and inverse diagonal C_*^{-1}. The generalized Bonferroni mean is given by,*

$$B_{\mathbb{M}}(\mathbf{x}) = C_*^{-1}\Big(M_1\Big(C\big(x_1, M_2(\mathbf{x}_{j\neq 1})\big), \ldots, C\big(x_n, M_2(\mathbf{x}_{j\neq n})\big)\Big)\Big). \quad (4)$$

The original Bonferroni mean is returned where $M_1 = WAM(\mathbf{x})$, $M_2 = PM_q(\mathbf{x})$ and $C = x^p y^q$ (with all weights equal).

Since M_1 is an averaging function of n arguments while M_2 is a function of $n-1$ arguments, they will have weighting vectors of different dimension. In order to choose the weights appropriately, so that they are consistent with the application and inputs, the following convention is used for the weighting vector of M_2 [8].

Given $\mathbf{u} \in [0,1]^n$, the vectors $\mathbf{u}^i \in [0,1]^{n-1}$, $i = 1, \ldots, n$ are defined by

$$u_j^i = \frac{u_j}{\sum_{k\neq i} u_k} = \frac{u_j}{1-u_i}, \quad u_i \neq 1. \quad (5)$$

Note that for every i, $\mathbf{u}^i$ sum to one.

This allows one to either use the same weighting vector or differing vectors if each stage of aggregation requires it.

2.2 Fitting aggregation functions to data

The usual framework for fitting a function f to data involves an objective equation that minimizes the difference between the observed values y_k and predicted values $f(\mathbf{x}_k)$ in some norm. In particular, we have the L_2 norm or least squares approach,

$$\sum_{k=1}^{K} \big(f(\mathbf{x}_k) - y_k\big)^2, \tag{6}$$

and L_1 or least absolute deviation (LAD) approach,

$$\sum_{k=1}^{K} \big|f(\mathbf{x}_k) - y_k\big|. \tag{7}$$

For our algorithms, we are interested in the latter approach, which can be converted into a linear program [2, 10].

Given a dataset with K rows $(x_{k1}, x_{k2}, ..., x_{kn}, y_k)$ where each k represents an observed value, we firstly represent each residual in terms of its positive and negative components (one of which will be zero), i.e. $r_k = |f(\mathbf{x})_k - y_k| = r_k^+ + r_k^-$.

We then minimize the sum of these residuals subject to equality constraints ensuring the y_k are equal to the predicted function value and the residual.

$$\begin{aligned} \text{Minimize} \quad & \sum_{k=1}^{K} r_k^+ + r_k^-, \\ \text{s.t.} \quad & f(\mathbf{x}_k) + r_k^+ - r_k^- = y_k, k = 1 \dots K \\ & r_k^+, r_k^- \geq 0. \end{aligned} \tag{8}$$

With suitable transformations or rearrangements of the data, many interesting aggregation functions can be represented in this way, for example, to fit a weighted quasi-arithmetic mean with gen-

erating function g, we can use the constraints:

$$\left(\sum_{i=1}^{n} w_i g(x_{ki})\right) + r_k^+ - r_k^- = g(y_k), k = 1 \ldots K$$
$$w_i \geq 0, \forall\ i,$$
$$\sum_{i=1}^{n} w_i = 1.$$

Note that the residuals in this case are the differences between the transformed data - not the actual data itself.

For ordered functions such as the OWA, the data can be transformed so that the weights are learned from the reordered data. In both cases, although the functions themselves are not linear, the weights are only fit to linear data.

In our AggWaFit.R source file, the commands `ordfit.GenOWA` and `ordfit.QAM` can be used to find the weighting vector $\mathbf{w}$ from a given data set where the generator and its inverse are specified. These commands are designed for fitting to data where the outputs are ordinal and will return a stats file with root mean squared error (RMSE), average L_1 loss, prediction accuracy (for predicting the ordinal classes), a confusion matrix and the resulting w. A file with the predicted values from the function and corresponding classes is also returned with the original data.

3 Formulating weights identification problems

The generalized Bonferroni mean is defined with respect to the two weighting vectors, $\mathbf{w}$ and $\mathbf{u}$. Due to its composition, however, we cannot transform the data and fit the weights as we do for the quasi-arithmetic mean and OWA. We look at three simplifications that will enable us to fit generalized Bonferroni means to data.

3.1 Fitting v_{ij} weights to product pairs

We can firstly consider the case of M_1, M_2 weighted arithmetic means and C the product operation with powers p, q. This leads to the

following expression and simplification.

$$\left(\sum_{i=1}^{n} w_i x_i^p \left(\sum_{j\neq i} \frac{u_j}{1-u_i} x_j^q\right)\right)^{\frac{1}{p+q}} = \left(\sum_{i=1,j=1,i\neq j}^{n} \frac{w_i u_j}{1-u_i} x_i^p x_j^q\right)^{\frac{1}{p+q}}$$

Although we still cannot separate the weights linearly, we can consider each $x_i x_j$ term and consider coefficients v_{ij}. We hence transform the instances of the dataset $(x_{k1}, x_{k2}, \ldots, x_{kn}, y_k)$ such that we fit to $(x_{k1}^p x_{k2}^q,\ x_{k1}^p x_{k3}^q, \ldots, x_{k,(n-1)}^p x_{kn}^q,\ y_k^{p+q})$ and introduce the following linear constraints.

$$\left(\sum_{i=1,j=1,i\neq j}^{n} v_{ij} x_{ki}^p x_{kj}^q\right) + r_k^+ - r_k^- = y_k^{p+q}, k = 1 \ldots K$$

$$v_{ij} \geq 0, \forall\ ij,$$

$$\sum_{i=1}^{n} v_{ij} = 1.$$

The resulting v_{ij} will not be separable into the w_i, u_i, u_j etc, however we can gain an idea of the rough contribution of w_i which is associated with the p index and u_i associated with the q index by summing the rows and columns of the v_{ij} matrix respectively.

In our BonFit.R source file, this fitting is done to ordinal data using the `ordfit.bonf.vij` command. Different p, q can be specified and further restrictions placed on the v_{ij} if desired.

3.2 Fitting w_i weights with fixed u

An alternative to fitting to the pairs $x_i^p x_j^q$ is to fix the weighting vector $\mathbf{u}$. This way, we can use alternative means for M_1, M_2 (in particular, any QAM) whereas before we were limited to weighted arithmetic means. We hence perform fitting by transforming each of the input terms x_{ki} by combining with the mean of the $\mathbf{x}_{k,j\neq i}$ and using the generator functions. Denoting the generator of M_1 by m_1, each of the terms will be given by

$$m_1\left(x_{ki}^p (M_2(\mathbf{x}_{k,j\neq i}))^q\right),$$

where the weighting vector for M_2 is defined separately for each i using each of the $\mathbf{u}^i$ determined from the supplied vector $\mathbf{u}$. In this case, we introduce the following constraints.

$$\left(\sum_{i=1,j=1,i\neq j}^{n} w_i m_1(x_{ki}^p(M_2(\mathbf{x}_{k,j\neq i}))^q)\right) + r_k^+ - r_k^- = (m_1(y_k))^{p+q}, \quad k = 1 \dots K,$$
$$w_i \geq 0, \qquad \forall\ i,$$
$$\sum_{i=1}^{n} w_i = 1.$$

This fitting is done with BonFit.R using `ordfit.bonf.quasi` where the generators of both M_1, M_2 can be specified, as well as the weighting vector $\mathbf{u}$ associated with M_2 and the indices p, q.

3.3 Enforcing mandatory requirements

In some applications, it may be desirable to define a model with one or two mandatory requirements, but which still fits the data as well as it can. In this case, we can use projections on $\mathbf{w}$. Denoting the generator of M_2 (a weighted quasi-arithmetic mean) by m_2, this will result in the following expression,

$$\left(x_i^p m_2^{-1}\left(\sum_{j\neq i}\frac{u_j}{1-u_i} m_2(x_j)\right)^q\right)^{\frac{1}{p+q}}.$$

As we can see, the u_i, u_j do not occur as linear cofactors, however since the i-th variable will always be mandatory, we can transform the dataset such that we only fit the weighting vector $\mathbf{u}^i$. We hence will not obtain a weight for the relative importance of i with respect to the other variables, however this would usually be acceptable as it is not needed in the model to calculate new values. By rearranging the function, we then introduce the following linear constraints into

the fitting problem.

$$\left(\sum_{j=1,j\neq i}^{n} u_j m_2(x_{kj})\right) + r_k^+ - r_k^- = m_2\left(\frac{y_k^{(p+q)/q}}{x_{ki}^{p/q}}\right),\ k = 1\ldots K,$$
$$u_j \geq 0, \forall\ j,$$
$$\sum_{j=1,j\neq i}^{n} u_j = 1.$$

To fit a function this way in BonFit.R, we can use the command `ordfit.bonf.proj`, where the variable that is to be mandatory is specified, as well as the desired generator for M_2.

4 Modeling the journal rankings dataset

In our previous work [3–5], we have used a dataset synthesized from the Australian journal rankings, which pairs the indices provided by Thomson and Reuters' ISI Web of Knowledge database with the quality ranks allocated by the Australian Research Council (ARC). The motivation for using such a dataset is that the relationship between the indices and the quality rank should be roughly monotone, while there are also likely to exist correlations between the inputs.

Before the rankings were disbanded, we collected the 2011 data for journals in all disciplines with both ARC and ISI data. This gave us a list of over 5000 journals spread across different fields of research (FoR) categories. For comparing the accuracy of the Bonferroni mean, we used 17 FoR categories, each with more than 80 journals and one (1103 Clinical Sciences) with 706 journals. The data first had to be transformed so that each of the indices ranged between $[0, 1]$ and so that the distribution of the scores was such that idempotency could be obtained. The algorithm from this is also available at the previously mentioned website. We used the algorithms in BonFit.R for the Bonferroni means ($m_2 = t^3$ for `ordfit.bonf.quasi`, and $m_2 = t$, $i = 2$, for `ordfit.bonf.proj` i.e. the Impact Factor is made mandatory), AggWaFit.R for the WAM, OWA, power means ($p = 2, 3$) and geometric mean, and *fmtools* for the Choquet integrals (2-additive and general). Table 1 shows the overall standard and 10-fold cross validation accuracy when fitting to the journals

Table 1. Overall classification and L1-accuracy for various aggregation functions

	$B.vij$	$B.qam$	$B.proj$	WAM	OWA	PM_2	PM_3	GM	Ch_{2-add}	Ch_{gen}
All										
L1	0.124	0.123	0.150	0.123	0.125	0.117	0.117	0.149	0.113	**0.106**
Acc.	0.676	0.662	0.576	0.661	0.642	0.672	0.673	0.621	0.691	**0.715**
10fold										
L1	0.132	0.126	0.170	0.129	0.133	**0.123**	0.124	0.216	0.126	0.126
Acc.	0.652	0.654	0.440	0.645	0.616	**0.655**	0.646	0.485	0.649	0.654

data, averaged across the 17 FoR codes. As we can see, the Bonferroni mean performs reasonably well in fitting to each of the datasets. Weighting each pair using v_{ij} could be interpreted similarly to modeling interaction between pairs as is done with the 2-additive Choquet integral, and it is worth noting that their performance is similar for the 10-fold tests. Enforcing the impact factor as a mandatory requirement in this case did not lead to good accuracy, however this may be necessary in some cases for reflecting the decision maker's preferences.

5 Conclusion

We have introduced some methods for fitting the generalized Bonferroni mean to data. To date, such methods have not been investigated for the Bonferroni mean. As well as making these available, we also draw attention to the datasets and R-code at our website, which can be used to further the study of aggregation functions and their use in decision making. We found that the generalized Bonferroni mean offers comparable performance to other means when modeling data. Although it is not possible to develop linear or quadratic programs in general, it is possible to write efficient algorithms for various special cases.

References

1. G. Beliakov. `fmtools` package, version 1.0, 2007, http://www.deakin.edu.au/∼gleb/aotool.html.
2. G. Beliakov. Construction of aggregation functions from data using linear programming. *Fuzzy Sets and Systems*, 160:65–75, 2009.

3. G. Beliakov and S. James. Citation based journal rankings: an application of fuzzy measures. In *Proc. of the 5th Intl Summer School of Aggregation Operators AGOP'09, July 6-11*, Palma de Mallorca, Spain, 2009.
4. G. Beliakov and S. James. Using Choquet integrals for evaluating citation indices in journal ranking. In *Proc. Eurofuse Workshop '09, September 16-18*, Pamplona, Spain, 2009.
5. G. Beliakov and S. James. Citation-based journal ranks: The use of fuzzy measures. *Fuzzy Sets and Systems*, 167:101–119, 2011.
6. G. Beliakov and S. James. Defining Bonferroni means over lattices. In *Proc. of FUZZIEEE*, Brisbane, Austarlia, 2012.
7. G. Beliakov and S. James. On extending generalized Bonferroni means to Atanassov orthopairs in decision making contexts. *Fuzzy Sets and Systems*, in press, DOI: 10.1016/j.fss.2012.03.018, 2012.
8. G. Beliakov, S. James, J. Mordelová, T. Rückschlossová, and R.R. Yager. Generalized Bonferroni mean operators in multi-criteria aggregation. *Fuzzy Sets and Systems*, 161:2227–2242, 2010.
9. G. Beliakov, A. Pradera, and T. Calvo. *Aggregation Functions: A Guide for Practitioners*. Springer, Heidelberg, 2007.
10. P. Bloomfield and W.L. Steiger. *Least Absolute Deviations. Theory, Applications and Algorithms*. Birkhauser, Boston, Basel, Stuttgart, 1983.
11. C. Bonferroni. Sulle medie multiple di potenze. *Bollettino Matematica Italiana*, 5:267–270, 1950.
12. J.J. Dujmovic. Continuous preference logic for system evaluation. *IEEE Trans. on Fuzzy Systems*, 15:1082–1099, 2007.
13. M. Grabisch, I. Kojadinovic, and P. Meyer. A review of methods for capacity identification in Choquet integral based multi-attribute utility theory: Applications of the Kappalab R package. *European Journal of Operational Research*, 186:766–785, 2008.
14. M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap. *Aggregation Functions*. Cambridge University Press, Cambridge, 2009.
15. J. Mordelová and T. Rückschlossová. ABC-aggregation functions. In *Proc. of the 5th Intl. Summer School on Aggregation Operators*, Palma de Mallorca, Spain, 2009.
16. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. http://www.R-project.org/.
17. Y. Torra and V. Narukawa. *Modeling Decisions. Information Fusion and Aggregation Operators*. Springer, 2007.
18. M. Xia, Z. Xu, and B. Zhu. Generalized intuitionistic fuzzy Bonferroni means. *International Journal of Intelligent Systems*, 27:23–47, 2011.
19. Z. Xu. Uncertain Bonferroni mean operators. *International Journal of Computational Intelligence Systems*, 3(6):761–769, 2010.
20. Z. Xu and R.R. Yager. Intuitionistic fuzzy Bonferroni means. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41:568–578, 2011.
21. R. Yager. On generalized Bonferroni mean operators for multi-criteria aggregation. *International Journal of Approximate Reasoning*, 50:1279–1286, 2009.
22. R. Yager, G. Beliakov, and S. James. On generalized Bonferroni means. In *Proc. Eurofuse 2009 Conference*, Pamplona, Spain, 2009.
23. R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Trans. on Systems, Man and Cybernetics*, 18:183–190, 1988.