# Fitting Aggregation Functions to Data: Part I - Linearization and Regularization

Maciej Bartoszuk[1], Gleb Beliakov[2], Marek Gagolewski[1,3(✉)], and Simon James[2]

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warsaw, Poland
bartoszukm@mini.pw.edu.pl, gagolews@ibspan.waw.pl
[2] School of Information Technology, Deakin University, 221 Burwood Hwy, Burwood, VIC 3125, Australia
{gleb,sjames}@deakin.edu.au
[3] Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland

**Abstract.** The use of supervised learning techniques for fitting weights and/or generator functions of weighted quasi-arithmetic means – a special class of idempotent and nondecreasing aggregation functions – to empirical data has already been considered in a number of papers. Nevertheless, there are still some important issues that have not been discussed in the literature yet. In the first part of this two-part contribution we deal with the concept of regularization, a quite standard technique from machine learning applied so as to increase the fit quality on test and validation data samples. Due to the constraints on the weighting vector, it turns out that quite different methods can be used in the current framework, as compared to regression models. Moreover, it is worth noting that so far fitting weighted quasi-arithmetic means to empirical data has only been performed approximately, via the so-called linearization technique. In this paper we consider exact solutions to such special optimization tasks and indicate cases where linearization leads to much worse solutions.

**Keywords:** Aggregation functions · Weighted quasi-arithmetic means · Least squares fitting · Regularization · Linearization

## 1 Introduction

In various situations, one is faced with a need to combine $n \geq 2$ numeric values in the unit interval, so that a single representative output is produced. Usually, some idempotent aggregation function, see, e.g., [7,13], is the required data fusion tool.

**Definition 1.** *We say that* $\mathsf{F} : [0,1]^n \to [0,1]$ *is an idempotent aggregation function, whenever it is nondecreasing in each variable, and for all* $x \in [0,1]$ *it holds* $\mathsf{F}(x,\ldots,x) = x$.


J.P. Carvalho et al. (Eds.): IPMU 2016, Part II, CCIS 611, pp. 767–779, 2016.
DOI: 10.1007/978-3-319-40581-0_62

Among useful idempotent aggregation functions we find weighted quasi-arithmetic means.

**Definition 2.** *Let $\varphi : [0,1] \to \bar{\mathbb{R}}$ be a continuous and strictly monotonic function and $\mathbf{w}$ be a weighting vector of length $n$, i.e., one such that for all $i$ it holds $w_i \geq 0$ and $\sum_{i=1}^{n} w_i = 1$. Then a* weighted quasi-arithmetic mean *generated by $\varphi$ and $\mathbf{w}$ is an idempotent aggregation function $\mathsf{WQAMean}_{\varphi,\mathbf{w}} : [0,1]^n \to [0,1]$ given for $\mathbf{x} \in [0,1]^n$ by:*

$$\mathsf{WQAMean}_{\varphi,\mathbf{w}}(\mathbf{x}) = \varphi^{-1}\left(\sum_{i=1}^{n} w_i \varphi(x_i)\right) = \varphi^{-1}\left(\mathbf{w}^T \varphi(\mathbf{x})\right).$$

Here are a few interesting cases in the above class:

- $\mathsf{WAMean}_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i = \mathbf{w}^T \mathbf{x}$, (weighted arithmetic mean, convex combination of inputs, $\varphi(x) = x$)
- $\mathsf{WHMean}_{\mathbf{w}}(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n} w_i / x_i}$, (weighted harmonic mean, $\varphi(x) = 1/x$)
- $\mathsf{WGMean}_{\mathbf{w}}(\mathbf{x}) = \prod_{i=1}^{n} x_i^{w_i}$, (weighted geometric mean, $\varphi(x) = \log x$)
- $\mathsf{PMean}_{r,\mathbf{w}}(\mathbf{x}) = \left(\sum_{i=1}^{n} w_i x_i^r\right)^{1/r}$ for some $r \neq 0$, (weighted power mean, $\varphi(x) = x^r$)
- $\mathsf{EMean}_{\gamma,\mathbf{w}}(\mathbf{x}) = \frac{1}{\gamma} \log\left(\sum_{i=1}^{n} w_i e^{\gamma x_i}\right)$ for some $\gamma \neq 0$. (weighted exponential mean, $\varphi(x) = e^{\gamma x}$)

Let us presume that we observe $m \geq n$ input vectors $\mathbf{X} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}] \in [0,1]^{n \times m}$ together with $m$ desired output values $\mathbf{Y} = [y^{(1)}, \ldots, y^{(m)}] \in [0,1]^{1 \times m}$ and that we would like to fit a *model* that determines the *best* functional relationship between the input values and the desired outputs. Generally, such a task is referred to as regression in machine learning. Nevertheless, classical regression models do not guarantee any preservation of important algebraic properties like the mentioned nondecreasingness or idempotence. Therefore, in our case, for a fixed generator function $\varphi$, we shall focus on the task concerning fitting a weighted quasi-arithmetic mean $\mathsf{WQAMean}_{\varphi,\mathbf{w}}$, compare, e.g., [4,7,12,21], to an empirical data set.

Given a *loss function* $E : \mathbb{R}^m \to [0,\infty)$ that is strictly decreasing towards $\mathbf{0}$, $E(0,\ldots,0) = 0$, the task of our interest may be expressed as an optimization problem:

$$\text{minimize } E\left(\varphi^{-1}\left(\sum_{i=1}^{n} w_i \varphi(x_i^{(1)})\right) - y^{(1)}, \ldots, \varphi^{-1}\left(\sum_{i=1}^{n} w_i \varphi(x_i^{(m)})\right) - y^{(m)}\right)$$

with respect to $\mathbf{w}$, under the constraints that $\mathbf{1}^T\mathbf{w} = 1$ and $\mathbf{w} \geq \mathbf{0}$. Typically, $E$ is an $L_p$ norm, in particular: $E(e_1,\ldots,e_m) = \sqrt{\sum_{i=1}^{m} e_i^2}$ (least squared error fitting, LSE), $E(e_1,\ldots,e_m) = \sum_{i=1}^{m} |e_i|$ (least absolute deviation fitting, LAD), or $E(e_1,\ldots,e_m) = \bigvee_{i=1}^{m} |e_i|$ (least maximum absolute deviation fitting, LMD).

In the weighted arithmetic mean case ($\varphi(x) = x$), it is well-known that an LSE fit can be expressed as a quadratic programming task, and both LAD and

LMD fits may be solved by introducing a few auxiliary variables and then by applying some linear programming solvers, see [6,9,12].

More generally, for arbitrary but fixed $\varphi$ (note that if $\varphi$ is unknown one may rely on a notion of spline functions to model a generator function, see [2,3,5,6,9,10]), the weight fitting task has up to now been solved approximately via a technique called *linearization*, compare [5,6,8,9,19]. Observing that if $y^{(j)}$ is not subject to any measurement error, i.e., we have $\varphi^{-1}\left(\sum_{i=1}^{n} w_i \varphi(x_i^{(j)})\right) = y^{(j)}$, instead of minimizing a function of $\varphi^{-1}\left(\sum_{i=1}^{n} w_i \varphi(x_i^{(j)})\right) - y^{(j)}$ we may consider a function of $\sum_{i=1}^{n} w_i \varphi(x_i^{(j)}) - \varphi(y^{(j)})$. Thus, the input and output values can be transformed prior to applying a weight fit procedure and then we may proceed in the same manner as when $\varphi(x) = x$ (and in fact deal with a linear interpolation problem). However, in practice this is rarely the case.

What is more, as noted recently in [12], we usually observe that a model may be overfit to a training data set and thus perform weakly on test or validation samples. Also, sometimes we would like to fit a function which is nondecreasing and idempotent but the input values need to be properly normalized prior to aggregate them so that these two important properties are meaningful.

The aim of this two-part paper is to complement the mentioned results (and extend the preliminary outcomes listed in [12]) concerning fitting of weighted quasi-arithmetic means (weighted arithmetic means in particular). In Sect. 2 we discuss possible ways to fit a weighted quasi-arithmetic mean without relying on the linearization technique. Moreover, we perform various numerical experiments that enable us to indicate in which cases linearization leads to a significant decrease in the fit quality. In Sect. 3 we discuss different ways of regularizing a model so as to prevent overfitting. One of the possible approaches consists of adding a special penalty, which is a common procedure in machine learning. However, as parameters of our model fulfill specific constraints (non-negative weights adding up to 1), other ways are possible in our setting too. Section 4 concludes this part of the contribution.

Moreover, in the second part [1] we deal with the problem of properly normalizing (transforming) discordant input values in such a way that idempotent aggregation functions may be fit. We present an application of such a procedure in a classification task dealing with the identification of pairs of similar R [17] source code chunks.

## 2 Linearization

From now on let us assume that $E(e_1, \ldots, e_m) = \sqrt{\sum_{i=1}^{m} e_i^2}$, i.e., we would like to find a least squares error fit. Such an approach is perhaps most common in machine learning literature [14]. What is more, most of the ideas presented in this paper can be quite easily applied in other settings.

Torra in [19,20] (compare also, e.g., [6,8,9]) already discussed weighted quasi-arithmetic mean fitting tasks. Nevertheless, it was noted that the problem is difficult in general, so one may simplify the problem assuming that the desired

outputs are not subject to errors. In such a case, noting that a fixed generator function $\varphi$ is surely invertible, we have for all $j$:

$$\sum_{i=1}^{n} w_i \varphi(x_i^{(j)}) = \varphi(y^{(j)}).$$

Using this assumption, instead of minimizing:

$$\|\varphi^{-1}\left(\mathbf{w}^T \varphi(\mathbf{X})\right) - \mathbf{Y}\|_2^2$$

one may decide to minimize a quite different (in general) goodness of fit measure:

$$\|\mathbf{w}^T \varphi(\mathbf{X}) - \varphi(\mathbf{Y})\|_2^2.$$

Such an approach is often called *linearization* of inputs.

Let us suppose, however, that we would like to solve the original weight fit problem and not the simplified (approximate) one.

*Example 1* ([12]). Suppose that $n = 5$ and we are given $m = 9$ toy data points given as below. Here $\mathbf{Y}$ was generated using $\mathbf{w} = (0.33, 0.43, 0.10, 0.08, 0.06)$ and $\varphi(x) = x^2$ with white noise was added ($\sigma = 0.05$).

| $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $x_1^{(j)}$ | 0.12 | 0.48 | 0.65 | 0.07 | 0.37 | 0.22 | 0.29 | 0.57 | 0.84 |
| $x_2^{(j)}$ | 0.73 | 0.41 | 0.45 | 0.79 | 0.92 | 0.23 | 0.90 | 0.40 | 0.57 |
| $x_3^{(j)}$ | 0.43 | 0.84 | 0.70 | 0.96 | 0.81 | 0.86 | 0.72 | 0.53 | 0.42 |
| $x_4^{(j)}$ | 0.52 | 0.75 | 0.48 | 0.40 | 0.62 | 0.28 | 0.80 | 0.92 | 0.79 |
| $x_5^{(j)}$ | 0.69 | 0.70 | 0.24 | 0.22 | 0.92 | 0.34 | 0.15 | 0.50 | 0.50 |
| $y^{(j)}$ | 0.65 | 0.58 | 0.70 | 0.51 | 0.82 | 0.56 | 0.70 | 0.64 | 0.75 |

Here are the true $\mathfrak{d}_1$, $\mathfrak{d}_2$, and $\mathfrak{d}_\infty$ error measures in the case of the linearized and the exact LSE and LAD minimization tasks.

| $E$ | $\mathfrak{d}_1$ | $\mathfrak{d}_2$ | $\mathfrak{d}_\infty$ |
|---|---|---|---|
| LAD – linearization | 0.7385 | 0.4120 | 0.2798 |
| LSE – linearization | 0.7423 | 0.2859 | 0.1626 |
| LAD – optimal | 0.7157 | 0.3170 | 0.2044 |
| LSE – optimal | 0.7587 | 0.2817 | 0.1501 |

In the above example, the differences are relatively small, but not negligible. While the use of the linearization technique for least-squared error fitting of quasi-arithmetic means will often lead to reliable results, there are clearly some situations where such a technique may not be justified. Fitting to the transformed dataset $\varphi(\mathbf{X}), \varphi(\mathbf{Y})$ essentially stretches the space along which the residuals are distributed and for some functions this will have a larger impact than others.

As an example, consider the geometric mean with generator $\varphi(x) = \log x$. For lower values of $y$, differences in the transformed residuals can become disproportionately large and pull the weights towards these data points. Further on we shall perform a few numerical experiments to indicate the generator functions that lead to much greater discrepancies.

### 2.1 Algorithms

We aim to:

$$\text{minimize} \sum_{j=1}^{m} \left( \varphi^{-1} \left( \sum_{i=1}^{n} w_i \varphi \left( x_i^{(j)} \right) \right) - y^{(j)} \right)^2 \quad \text{w.r.t. } \mathbf{w}$$

subject to $\mathbf{w} \geq \mathbf{0}$ and $\mathbf{1}^T \mathbf{w} = 1$. By homogeneity and triangle inequality of $\|\cdot\|_2$ we have that this is a convex optimization problem.

**A Solution Based on a Nonlinear Solver.** First of all, we may consider the above as a generic nonlinear optimization task. To drop the constraints on $\mathbf{w}$, we can use an approach considered (in a different context) by Filev and Yager [11], see also [20]. We take a different parameter space, $\boldsymbol{\lambda} \in \mathbb{R}^n$, such that:

$$w_i = \frac{\exp(\lambda_i)}{\sum_{k=1}^{n} \exp(\lambda_k)}.$$

Assuming that $\varphi^{-1}$ is differentiable, let us determine the gradient $\nabla E(\boldsymbol{\lambda})$. For any $k = 1, \ldots, n$ it holds:

$$\frac{\partial}{\partial \lambda_k} E(\boldsymbol{\lambda}) = 2 \frac{\exp(\lambda_k)}{\sum_{i=1}^{n} \exp(\lambda_i)} \sum_{j=1}^{m} \left( \varphi^{-1} \left( \frac{\sum_{i=1}^{n} \exp(\lambda_i) \varphi \left( x_i^{(j)} \right)}{\sum_{i=1}^{n} \exp(\lambda_i)} \right) - y^{(j)} \right)$$
$$\cdot (\varphi^{-1})' \left( \frac{\sum_{i=1}^{n} \exp(\lambda_i) \varphi \left( x_i^{(j)} \right)}{\sum_{i=1}^{n} \exp(\lambda_i)} \right) \cdot \left( \varphi \left( x_k^{(j)} \right) - \frac{\sum_{i=1}^{n} \exp(\lambda_i) \varphi \left( x_i^{(j)} \right)}{\sum_{i=1}^{n} \exp(\lambda_i)} \right).$$

Assuming that $\mathbf{Z} = \mathbf{w}^T \varphi(\mathbf{X})$ and $\mathbf{w} = \exp(\boldsymbol{\lambda}) / \mathbf{1}^T \exp(\boldsymbol{\lambda})$, we have:

$$\nabla E(\boldsymbol{\lambda}) = 2 \cdot \mathbf{w} \cdot \left( \left( \phi^{-1}(\mathbf{Z}) - Y \right) \cdot \left( \varphi^{-1} \right)' (\mathbf{Z}) \right) \times \left( \varphi(\mathbf{X})^T - \mathbf{Z} \right),$$

where $\cdot, -$ stand for elementwise vectorized multiplication and subtraction, respectively, $\times$ denotes matrix multiplication, and $\varphi(\mathbf{X})^T - \mathbf{Z}$ means that we subtract $\mathbf{Z}$ from each column in $\varphi(\mathbf{X})^T$. The solution may be computed using, e.g., a quasi-Newton nonlinear optimization method by Broyden, Fletcher, Goldfarb and Shanno (the BFGS algorithm, see [16]). However, let us note that while using the mentioned reparametrization, the BFGS algorithm may occasionally fail to converge as now the solution is not unique.

Another possible way of solving the above problem would be to rewrite the objective as a function of $n-1$ variables $v_1, \ldots, v_{n-1}$ such that $w_i = v_i$ for $i = 1, \ldots, n-1$ and $w_n = 1 - \sum_{i=1}^{n-1} v_i$. This is a constrained optimization problem, but the constraints on $\mathbf{v}$ are linear: $\mathbf{v} \geq \mathbf{0}$ and $\mathbf{1}^T\mathbf{v} \leq 1$. With generic nonlinear solvers, such an optimization task is usually determined by adding an appropriate barrier function (e.g., logarithmic barrier, see [16]) term to the objective function.

**Compensation Factors in Linearization.** Another possible way is to apply a compensation factor such that for any residual in the linearization technique $v^{(j)} = \left(\sum_{i=1}^{n} \varphi(x_i^{(j)})\right) - \varphi(y^{(j)})$, we estimate the true residual $r^{(j)} = \varphi^{-1}\left(\sum_{i=1}^{n} \varphi(x_i^{(j)})\right) - y^{(j)}$.

We begin with an estimate of our true residual, which we denote by $r_{\text{est}}$. The estimated residual for any known $v^{(j)}$ is then calculated as:

$$\text{est}(r^{(j)}) = \frac{v^{(j)} r_{\text{est}}}{\varphi(y^{(j)} + r_{\text{est}}) - \varphi(y^{(j)})}.$$

In other words, we calculate the average rate of change between $y^{(j)}$ and $y^{(j)} + r_{\text{est}}$ and then use the reciprocal of this as our compensation factor. A visual illustration of this process is shown in Fig. 1.

Where $y^{(j)} + r_{\text{est}}$ is outside our domain $[0, 1]$, we can instead use the average rate of change between $y^{(j)}$ and the boundary point (or very close to the boundary if $\varphi$ is infinite).

Obviously the average rate of change will differ depending on whether $v^{(j)}$ is positive or negative, and so we can split $v^{(j)}$ into its positive and negative components so that a different compensation factor can be applied.

We let $v^{(j)} = v_+^{(j)} + v_-^{(j)}$ where $v_+^{(j)} \geq 0, v_-^{(j)} \leq 0$ which we then use as decision variables in our quadratic programming task. We optimize with respect to these,

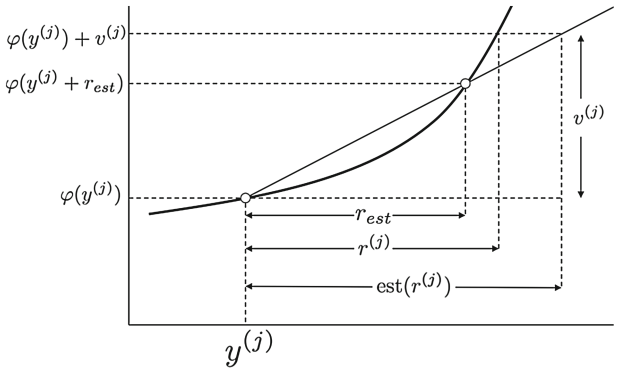


**Fig. 1.** Illustration of how compensation factor is calculated. From a generating function $\varphi$ we estimate $r^{(j)}$ using the average rate of change between $y^{(j)}$ and $y^{(j)} + r_{\text{est}}$.

and add constraints:

$$\left(\sum_{i=1}^{n} \varphi(x_i^{(j)})\right) + v_+^{(j)} + v_-^{(j)} = \varphi(y^{(j)}).$$

In summary, we have the following quadratic programming task.

$$\text{minimize} \sum_{j=1}^{m} \left(\frac{v_+^{(j)} r_{\text{est}}}{\varphi(y^{(j)}+r_{\text{est}})-\varphi(y^{(j)})}\right)^2 + \left(\frac{v_-^{(j)} r_{\text{est}}}{\varphi(y^{(j)}-r_{\text{est}})-\varphi(y^{(j)})}\right)^2 \quad \text{w.r.t. } \mathbf{w}, \mathbf{v}_-, \mathbf{v}_+$$

$$\text{such that} \quad \sum_{i=1}^{n} w_i = 1, w_i \geq 0, i = 1, \ldots, n,$$

$$\left(\sum_{i=1}^{n} \varphi(x_i^{(j)})\right) + v_+^{(j)} + v_-^{(j)} = \varphi(y^{(j)}),$$

$$v_+^{(j)} \geq 0, -v_-^{(j)} \geq 0, j = 1, \ldots, m.$$

Since the usefulness of this method will depend on $r_{\text{est}}$, we can also set up a bilevel optimization problem such that it is optimized for the given training set. Nevertheless, note that this time we deal with an approximate method. The following experiments show that the method is useful for compensating for the stretching effect of the generating function and also help us identify some specific instances of where linearization by itself has poor performance.

### 2.2 Experiments

For each of the generating functions $\varphi(x) = x^2, x^3, x^{1/2}, \log x$ we created data sets with $m = 20$ and $m = 100$ test points. For each trial, we generated $\mathbf{w}$ randomly and then after calculating the desired output values we added Gaussian noise with $\sigma = 0.05$ and 0.1. We then measured the LSE using:

(i) the linearization technique where the data are transformed using $\varphi(\mathbf{X}), \varphi(\mathbf{Y})$, i.e., the sum of $\left(\left(\sum_{i=1}^{n} \varphi(x_i^{(j)})\right) - \varphi(y^{(j)})\right)^2$,
(ii) the method proposed here with $r_{\text{est}} = 0.1$,
(iii) the method proposed here with $r_{\text{est}}$ optimized,
(iv) a general non-linear optimization solver.

The $\mathbf{x}^{(j)}$ vectors were generated both from the uniform distribution on $[0, 1]^n$ as well as an exponential distribution scaled to the interval $[0, 1]$. The uniform data would be expected to result in $y$ values distributed around the middle of the interval, while exponentially distributed data would often result in outputs closer to the lower end of the interval. We expect the latter case to result in worse performance for linearization.

After obtaining the fitted weighted vectors, we calculated the total LSE and normalized these values by expressing them as a proportion of the optimal LSE from the weighting vector obtained from the generalized solver. The results are shown in Table 1.

**Table 1.** Relative LSE calculated as proportion of total LSE obtained using a general nonlinear solver *(iv)*. Results represent averages over 10 trials

| Uniformly distributed $\mathbf{X}$ | | | | | | |
|---|---|---|---|---|---|---|
| $m = 20$ | $\sigma = 0.05$ | | | $\sigma = 0.1$ | | |
| $\varphi(x)$ | *(i)* | *(ii)* | *(iii)* | *(i)* | *(ii)* | *(iii)* |
| $x^2$ | 0.0623 | 0.0011 | 0.0005 | 0.0270 | 0.0159 | 0.0026 |
| $x^3$ | 0.0936 | 0.0114 | 0.0025 | 0.1329 | 0.1038 | 0.0131 |
| $x^{1/2}$ | 0.0233 | 0.0006 | 0.0004 | 0.0725 | 0.0099 | 0.0008 |
| $\log x$ | 0.1492 | 0.0042 | 0.0025 | 0.4763 | 0.0358 | 0.0055 |
| $m = 100$ | $\sigma = 0.05$ | | | $\sigma = 0.1$ | | |
| $\varphi(x)$ | *(i)* | *(ii)* | *(iii)* | *(i)* | *(ii)* | *(iii)* |
| $x^2$ | 0.0072 | 0.0004 | 0.0003 | 0.0106 | 0.0070 | 0.0015 |
| $x^3$ | 0.0266 | 0.0017 | 0.0014 | 0.0356 | 0.0499 | 0.0080 |
| $x^{1/2}$ | 0.0080 | 0.0002 | 0.0001 | 0.0061 | 0.0014 | 0.0005 |
| $\log x$ | 0.1921 | 0.0013 | 0.0011 | 0.2550 | 0.0064 | 0.0017 |
| Exponentially distributed $\mathbf{X}$ | | | | | | |
| $m = 20$ | $\sigma = 0.05$ | | | $\sigma = 0.1$ | | |
| $\varphi(x)$ | *(i)* | *(ii)* | *(iii)* | *(i)* | *(ii)* | *(iii)* |
| $x^2$ | 0.0479 | 0.0066 | 0.0035 | 0.0795 | 0.0946 | 0.0132 |
| $x^3$ | 0.1560 | 0.0234 | 0.0145 | 0.2345 | 0.3030 | 0.0266 |
| $x^{1/2}$ | 0.1048 | 0.0061 | 0.0036 | 0.0522 | 0.0204 | 0.0047 |
| $\log x$ | 1.0970 | 0.0133 | 0.0076 | 2.7748 | 0.0558 | 0.0194 |
| $m = 100$ | $\sigma = 0.05$ | | | $\sigma = 0.1$ | | |
| $\varphi(x)$ | *(i)* | *(ii)* | *(iii)* | *(i)* | *(ii)* | *(iii)* |
| $x^2$ | 0.0198 | 0.0042 | 0.0041 | 0.0273 | 0.0520 | 0.0055 |
| $x^3$ | 0.0579 | 0.0100 | 0.0040 | 0.0820 | 0.1641 | 0.0117 |
| $x^{1/2}$ | 0.0167 | 0.0015 | 0.0012 | 0.0204 | 0.0098 | 0.0045 |
| $\log x$ | 1.8401 | 0.0095 | 0.0049 | 0.7770 | 0.0460 | 0.0228 |

A number of observations can be made from this data. Firstly, we note that linearization for $\varphi(x) = x^2$ tended only to produce increases in LSE of about 2–7 %, regardless of the data distribution. On the other hand, $\varphi(x) = x^3$ showed increases in LSE of between 9–24 % when there were only $m = 20$ data instances.

The most dramatic results were obtained when fitting the geometric mean. For exponentially distributed input vectors, the method of linearization increased the error by up to 277 % when the noise added to $y$ was generated using $\sigma = 0.1$. There was large variability in these trials – in the best case using linearization increased the error by 45 %, while at other times the difference was 10 fold (LSE was 0.8752 compared with 0.0940 when $r_{\text{est}} = 0.1$ was used, about a 3.2 % increase on the optimal LSE). The worst results seemed to occur where the

data set included $y$ values equal to zero. The weight associated with the lowest input for that instance is pulled up to try and reduce the very large error. The compensation factor however did seem to obtain decent improvements even when the data was exponentially distributed.

While a setting of $r_{\text{est}} = 0.1$ tended to result in significant improvements for small errors ($\sigma = 0.05$), increased error of $\sigma = 0.1$ often had optimal values that were closer to 0.2.

## 3 Regularization

In this section we discuss a few possible ways to prevent a model being overfit to a training sample. In other words, we would like that for other samples of the same kind (e.g., following the same statistical distribution) the model performance does not decrease drastically.

### 3.1 Tikhonov Regularization

The Tikhonov regularization [18] is a basis for the ridge regression [15] method. It has a form of an additional penalty term dependent on a scaled squared $L_2$ norm of the weighting vector.

In our case we may consider, for some $\lambda$, an optimization task:

$$\text{minimize} \sum_{j=1}^{m} \left( \varphi^{-1} \left( \sum_{i=1}^{n} w_i \varphi(x_i^{(j)}) \right) - y^{(j)} \right)^2 + \lambda \sum_{i=1}^{n} w_i^2 \quad \text{w.r.t. } \mathbf{w}$$

subject to $\mathbf{1}^T\mathbf{w} = 1$ and $\mathbf{w} \geq \mathbf{0}$. Note that due to the usual constraints on $\mathbf{w}$, the use of the $L_1$ norm instead of squared $L_2$ (like, e.g., in Lasso regression) does not make much sense at this point: we always have $\|\mathbf{w}\|_1 = 1$.

In the simplest case ($\varphi(x) = x$), the above optimization problem can be written in terms of the following quadratic programming task:

$$\text{minimize } 0.5\,\mathbf{w}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})\mathbf{w} - (\mathbf{X}\mathbf{Y}^T)^T\mathbf{w} \quad \text{w.r.t. } \mathbf{w}$$

subject to $\mathbf{w} \geq \mathbf{0}$, $\mathbf{1}^T\mathbf{w} = 1$, which minimizes the squared error plus a $\lambda\|\mathbf{w}\|_2^2$ penalty term. Note that for other generator functions $\varphi$ we can easily incorporate an appropriate penalty to an optimization task considered in the previous section.

*Remark 1.* Unlike in regression problems, where we always presuppose that $\lambda \geq 0$, in our framework we are bounded with additional constraints on $\mathbf{w}$ which, for large $\lambda$, tend to generate weighting vectors such that $w_i \to 1/n$. On the other hand, in the current framework the case of $\lambda < 0$ may also lead to useful outcomes. Yet, we should note that for $\lambda \to -\infty$ we observe that $w_j \to 1$ for some $j$.

*Example 2.* Let us consider a data set generated randomly with R as follows:

```
set.seed(321)
n <- 10; m <- 100
realw <- rbeta(n, 1, 5)
realw <- realw/sum(realw) # real weights ~ beta distribution
X <- t(matrix(runif(n*m), nrow=m))
Y <- t(realw) %*% X + rnorm (m, 0, 0.1)
Y[,] <- pmax(0, pmin(1, Y))
X <- round(X, 2) # uniform distribution, rounded
Y <- round(Y, 2) # sigma=0.1, truncated to [0,1], rounded
train <- sample(1:m, m*0.8)
X_test <- X[,-train,drop=FALSE] # test sample
Y_test <- Y[,-train,drop=FALSE]
X <- X[,train,drop=FALSE]       # training sample
Y <- Y[,train,drop=FALSE]
```

The data points are divided into two groups: a training sample (80 % of the observations, used to estimate the weights) and a test sample (20 %, used to compute the error). Figure 2 depicts squared error measures as a function of Tikhonov regularization coefficient $\lambda$. We see we were able to improve the error measure by ca. 9 % by using $\lambda \simeq 4.83$.

| $E$ | $\mathfrak{d}_1$ | $\mathfrak{d}_2$ | $\mathfrak{d}_\infty$ |
|---|---|---|---|
| — (using `realw`) | 1.291 | 0.3637 | 0.1915 |
| LAD | 1.488 | 0.4027 | 0.1937 |
| LSE | 1.436 | 0.4036 | 0.1931 |
| LMD | 1.475 | 0.4025 | 0.1699 |
| LSE + regularization, $\lambda \simeq 4.83$ | 1.371 | 0.3705 | 0.1891 |

### 3.2 Weights Dispersion Entropy

Similar to minimizing sum of squared weights, maximizing weights dispersion entropy can also have benefits, measuring the degree to which the function takes into account all the inputs, compare [9] for its use for a different purpose. It is given by:

$$\mathsf{Disp}(\mathbf{w}) = -\sum_{i=1}^{n} w_i \log w_i,$$

with the convention $0 \cdot \log 0 = 0$.

In cases where fitting results in multiple solutions, for example when there is too few data for there to be a singular minimizer, Torra proposed the use of weights dispersion as an additional criterion to determine the best solution [19]. It is implemented as a second level of the optimization. After obtaining a

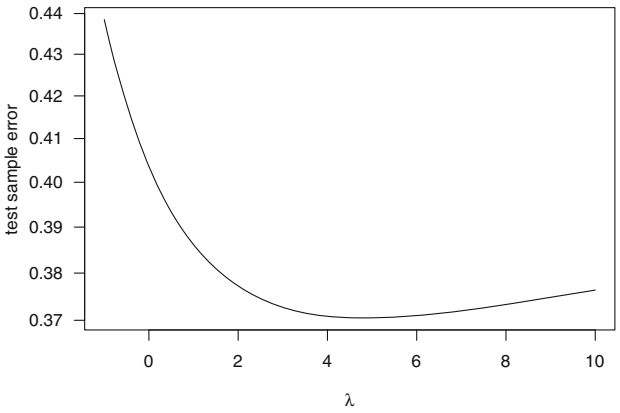


**Fig. 2.** Three error measures on a test data set from Example 2 as a function of regularization penalty $\lambda$.

minimum $A$ to the objective in the standard least squares fitting problem, one then solves:

$$\text{minimize} \sum_{i=1}^{n} w_i \log w_i + \lambda \left( \sum_{j=1}^{m} \left( \varphi^{-1} \left( \sum_{i=1}^{n} w_i \varphi(x_i^{(j)}) \right) - y^{(j)} \right)^2 - A \right)^2 \quad \text{w.r.t. } \mathbf{w}$$

$$\text{such that} \quad \sum_{i=1}^{n} w_i = 1, w_i \geq 0, i = 1, \ldots, n,$$

for some $\lambda > 0$.

One may alternatively consider a one-level task like:

$$\text{minimize} \sum_{j=1}^{m} \left( \varphi^{-1} \left( \sum_{i=1}^{n} w_i \varphi(x_i^{(j)}) \right) - y^{(j)} \right)^2 + \lambda \sum_{i=1}^{n} w_i \log w_i \quad \text{w.r.t. } \mathbf{w}$$

subject to the standard constraints.

*Remark 2.* In fact, weights dispersion and sum of squared weights are both examples of functions used to model income inequality in economics and evenness in ecology. There are numerous other functions used in these fields that could also be used as secondary objectives to achieve the task of weight regularization (e.g., the Gini index).

## 4 Conclusion

We have considered some practical issues concerning fitting weighted quasi arithmetic means to empirical data using supervised learning-like approaches. First

of all, we pointed out the drawbacks of the commonly applied linearization technique, which may lead to far-from-optimal solutions. Moreover, we analyzed some ways to prevent model overfitting.

Note that the discussion can be easily generalized to other error measures and fitting other aggregation functions parametrized via a weighting vector, e.g., weighted Bonferroni means and generalized OWA operators. In future applications, the compensation and regularization techniques proposed here can be used to learn more useful and informative models.

**Acknowledgments.** This study was supported by the National Science Center, Poland, research project 2014/13/D/HS4/01700.

## References

1. Bartoszuk, M., Beliakov, G., Gagolewski, M., James, S.: Fitting aggregation functions to data: Part II - idempotization. In: Carvalho, J.P., Lesot, M.-J., Kaymak, U., Vieira, S., Bouchon-Meunier, B., (eds.) IPMU 2016, Part II, CCIS, vol. 611, pp. 780–789. Springer, Heidelberg (2016)
2. Beliakov, G.: Shape preserving approximation using least squares splines. Approximation Theory Appl. **16**(4), 80–98 (2000)
3. Beliakov, G.: Monotone approximation of aggregation operators using least squares splines. Int. J. Uncertainty Fuzziness Knowl. Based Syst. **10**, 659–676 (2002)
4. Beliakov, G.: How to build aggregation operators from data. Int. J. Intell. Syst. **18**, 903–923 (2003)
5. Beliakov, G.: Learning weights in the generalized OWA operators. Fuzzy Optim. Decis. Making **4**, 119–130 (2005)
6. Beliakov, G.: Construction of aggregation functions from data using linear programming. Fuzzy Sets Syst. **160**, 65–75 (2009)
7. Beliakov, G., Bustince, H., Calvo, T.: A Practical Guide to Averaging Functions. STUDFUZZ. Springer, Switzerland (2016)
8. Beliakov, G., James, S.: Using linear programming for weights identification of generalized bonferroni means in R. In: Narukawa, Y., López, B., Villaret, M., Torra, V. (eds.) MDAI 2012. LNCS, vol. 7647, pp. 35–44. Springer, Heidelberg (2012)
9. Beliakov, G., Pradera, A., Calvo, T.: Aggregation Functions: A Guide for Practitioners. Springer, Heidelberg (2007)
10. Beliakov, G., Warren, J.: Appropriate choice of aggregation operators in fuzzy decision support systems. IEEE Trans. Fuzzy Syst. **9**(6), 773–784 (2001)
11. Filev, D., Yager, R.R.: On the issue of obtaining OWA operator weights. Fuzzy Sets Syst. **94**, 157–169 (1998)
12. Gagolewski, M.: Data Fusion: Theory, Methods, and Applications. Institute of Computer Science, Polish Academy of Sciences, Warsaw (2015)
13. Grabisch, M., Marichal, J.L., Mesiar, R., Pap, E.: Aggregation Functions. Cambridge University Press, New York (2009)
14. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2013)
15. Hoerl, A., Kennard, R.: Ridge regression: biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970)

16. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (2006)
17. R Development Core Team: R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2016). http://www.R-project.org
18. Tikhonov, A., Arsenin, V.: Solution of Ill-posed Problems. Winston & Sons, Washington (1977)
19. Torra, V.: Learning weights for the quasi-weighted means. IEEE Trans. Fuzzy Syst. **10**(5), 653–666 (2002)
20. Torra, V.: OWA operators in data modeling and reidentification. IEEE Trans. Fuzzy Syst. **12**(5), 652–660 (2004)
21. Torra, V.: Aggregation operators and models. Fuzzy Sets Syst. **156**, 407–410 (2005)