Least Median of Squares (LMS) and Least Trimmed Squares (LTS) Fitting for the Weighted Arithmetic Mean

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

[1] School of Information Technology, Deakin University, Burwood, Victoria, Australia
{gleb,sjames}@deakin.edu.au

[2] Systems Research Institute, Polish Academy of Sciences,
ul. Newelska 6, 01-447 Warsaw, Poland
gagolews@ibspan.waw.pl

[3] Faculty of Mathematics and Information Science,
Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warsaw, Poland

Abstract. We look at different approaches to learning the weights of the weighted arithmetic mean such that the median residual or sum of the smallest half of squared residuals is minimized. The more general problem of multivariate regression has been well studied in statistical literature, however in the case of aggregation functions we have the restriction on the weights and the domain is also usually restricted so that 'outliers' may not be arbitrarily large. A number of algorithms are compared in terms of accuracy and speed. Our results can be extended to other aggregation functions.

Keywords: Aggregation functions · Robust statistics
Least median of squares fitting · Least trimmed squares fitting

1 Introduction

In the application of aggregation functions, a key problem is how to determine the weights or function parameters that give the best fit with respect to some penalty or objective to an observed dataset. The learned parameters can then be used either for data analysis or in the prediction of new values.

A standard approach is to use programming methods such that the sum of residuals is optimized [1–4], e.g. for a weighted arithmetic mean with respect to an unknown n-dimensional vector of weights $\mathbf{w}$, and an observed dataset consisting of m input-output pairs $(\mathbf{x}_i, y_i), \mathbf{x}_i \in [0,1]^n, y_i \in [0,1]$, we have

J. Medina et al. (Eds.): IPMU 2018, CCIS 854, pp. 367–378, 2018.
https://doi.org/10.1007/978-3-319-91476-3_31

$$
\begin{aligned}
\underset{\mathbf{w}}{\text{Minimize}} \quad & \sum_{i=1}^{m} |r_i|^p && (1)\\
\text{s.t.} \quad & r_i = \left(\sum_{j=1}^{n} w_j x_{ij} \right) - y_i, \qquad i = 1, \dots, m,\\
& \sum_{j=1}^{n} w_j = 1, \qquad w_j \geq 0, \qquad j = 1, \dots, n.
\end{aligned}
$$

For $p = 2$ we are minimizing the sum of squared residuals or least squares (LS), which can be solved as a quadratic programming problem, while for $p = 1$, we have the least absolute deviation (LAD), which can be solved using linear programming methods by introducing two decision variables for each observed instance and setting $r_i = r_i^+ - r_i^-$ and $r_i^+, r_i^- \geq 0$.

The LAD approach should be less susceptible to outliers, however as has been well observed in statistical literature [5], leverage points can still exert influence if the residual associated with outliers is significantly larger than residuals associated with other points.

For example, consider the 2-variate data depicted in blue in Fig. 1(a).

When fitting using Eq. (1) and $p = 1$ (LAD[1]), we obtain the weighting vector $\mathbf{w} = (0.300, 0.700)$ with a total fitting error of 4.9×10^{-4}. Using $p = 2$ (LS), we also obtain a good result with the same weighting vector (to 3 d.p.) and error 2.8×10^{-8}.

However suppose we introduce outlying points at $\mathbf{x} = (0, 1)$, $y = 0$. These are indicated by the red point depicted in Fig. 1(a). With the introduction of a single outlier, the LS results in $\mathbf{w} = (0.740, 0.260)$, the penalty or objective value increasing to 1.3×10^{-1} (note that these weights reverse the importance allocated to each variable). The outlier effect on the LS method is illustrated visually in Fig. 1(b)–(c). With the single outlier, the weights determined by LAD are almost unchanged (when evaluated to 3 d.p.), the error increasing to 7.0×10^{-1}. However when we introduce 2 outliers (at the same point), the LS continues to allocate more weight to the first variable, $\mathbf{w} = (0.841, 0.159)$ with overall penalty 2.2×10^{-1} and, at this point, the LAD fitting results in the vector $\mathbf{w} = (1, 0)$, i.e. interpolating the two outliers, because the sum of the residuals when fitting to these points is 1.4, which is less than the error that would result if the original model's weighting vector $\mathbf{w} = (0.7, 0.3)$ were used.

In the 80s, this problem for standard linear regression prompted Rousseeuw and others [5,7,8] to consider optimizing with respect to the median residual (least median of squares or LMS) or the sum of the smallest 50% of residuals (least trimmed squares or LTS) instead, i.e. minimizing $\left|r_{(k)}\right|^p$ or $\sum_{i=1}^{k} |r_{(k)}|^p$ where $k = \lceil m/2 \rceil$ and $|r_{(i)}|$ indicates the i-th smallest residual.

[1] All fitting performed in R [6] with details available at http://aggregationfunctions.wordpress.com.

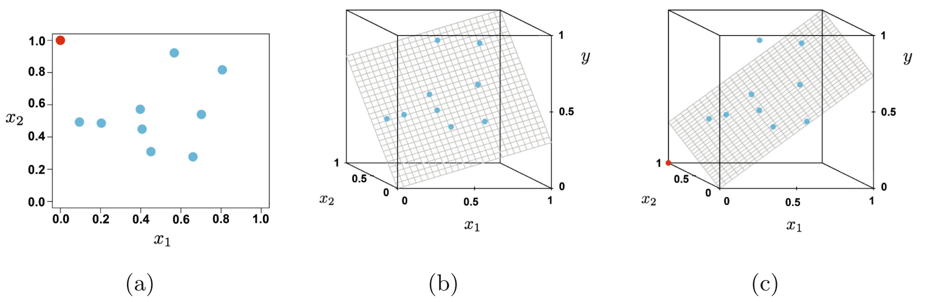

Fig. 1. (a) Randomly generated data (uniformly over $[0,1]^2$) in blue and an outlying point in red shown as projection onto the 2-dimensional plane. (b)–(c) Data from (a) with a well fitting weighted mean (b) and a weighted mean affected by the outlier in red (c), both determined using least squares fitting. In the latter case a single outlier pulls the function towards the outlying point and in 3-dimensional space. (Color figure online)

In [5], Rousseeuw notes that the breakdown point, i.e. the percentage of data that can be arbitrarily large before a reliable result is obtained, is $((m/2)-n+2)/m$.

Rousseeuw's method involves sampling n points (or $n+1$ in the case of standard regression requiring an intercept), solving the exact interpolation problem, then checking the residuals. After multiple iterations, the weighting vector that minimizes the objective function of the residuals is taken as the approximate solution. The number of samples can be chosen such that the probability of a 'good' solution appearing in one of the samples is high. An underlying assumption then is that there exists a sample of n points that are representative enough of the non-outlier dataset. Rousseeuw also has investigated the reliability in terms of estimating accuracy assuming normally distributed error.

In the case of weighted means, solving the interpolation problem for n points could result in negative weights if the data contains noise, and depending on the granularity at which data is collected, real data is likely to include subsets of observed points resulting in singular matrices and hence be unsolvable. It is noted in [9] that minimizing the median residual has a relationship to the infinity norm (L_∞), i.e. the problem can be expressed as a mixed integer program where the maximum residual is minimized for a subset consisting of half of the data (which theoretically could be implemented using binary variables indicating whether a datum is included or not). Of course, for any reasonable sized dataset this quickly becomes infeasible, however we can still look to use the minimization of the maximum residual as the basis of a number of approximation algorithms. Furthermore, it should be noted that with computing power and the availability of general-purpose solvers, many real applications would have the luxury of being able to spend a little extra computing time if high accuracy is needed, so a range of approaches are practically feasible.

In this contribution, we introduce and investigate a number of algorithms that aim to find the best approximation to the weights of a weighted arithmetic mean that minimize the LMS and LTS fitting criteria. We test the algorithms against synthetic data to determine whether their respective performance is dependent on factors such as the number of outliers, the structure of the outliers, and the variable parameters of each algorithm. In Sect. 2 we give a brief overview of aggregation functions (of which the weighted arithmetic mean is an archetypical example) and the data-fitting problem. In Sect. 3 a number of algorithms are presented and compared with numerical experiments. Some concluding remarks are provided in the final section.

2 Preliminiaries

We are concerned with the modelling of data with aggregation functions [1,3,4,10,11], a class of multi-variate functions $A : [0,1]^n \to [0,1]$ satisfying monotonicity in each argument and boundary conditions $A(0,\ldots,0) = 0$ and $A(1,\ldots,1) = 1$.

Although a broad definition, in the context of machine learning, the monotonicity of aggregation functions ensures a degree of robustness and conceptual reliability in the obtained model (provided monotonicity makes sense in the application), while the boundary conditions to some extent ensure that the scale of the output can be interpreted over a similar scale to the inputs. In particular, we will focus on use of the weighted arithmetic mean, $\text{WAM}(\mathbf{x}) = \sum\limits_{j=1}^{n} w_i x_i$, with $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ an n-dimensional weighting vector that satisfies $\sum_{i=1}^{n} w_i = 1$ and $w_i \geq 0, \forall i$.

The weighted arithmetic mean is said to be averaging, i.e. for all $\mathbf{x} \in [0,1]^n$, $\min(\mathbf{x}) \leq \text{WAM}(\mathbf{x}) \leq \max(\mathbf{x})$.

There are countless families of aggregation functions with various interesting properties, including those that are averaging and defined with respect to weighting vectors. While we focus on the simplest family, most of our results would be easily extended to the cases of OWA operators, weighted quasi-arithmetic means and the Choquet integral to name a few. We note too that other intervals can be considered, however we will contain ourselves to $[0,1]$ here.

How to fit weighted arithmetic means to data based on least absolute deviation has been addressed in [2,12–14]. We recall that Eq. (1) can be used as the basis for finding the best fitting aggregation function, while further requirements on the weights may also be desired (see, e.g. the summaries and references in [1]).

More complicated aggregation functions can be fit to data using more or less the same approach. Ordered weighted averaging (OWA) functions merely require each of the input vectors to be sorted, while the fitting can be performed on weighted quasi-arithmetic means by transforming the inputs and outputs (although this can result in residuals being over- or under-estimated, see [15]).

3 Least Median of Squares (LMS) and Least Trimmed Squares (LTS) Fitting for the Weighted Arithmetic Mean

The difficult aspect of solving Eq. (1) with respect to the LMS or LTS is determining the subset $S \subset \{1, \ldots, m\}$ such that $|S| = \lceil m/2 \rceil$ and there exists an observation k with $|r_k| \geq |r_i|, \forall\, i \in S$ and $|r_k| \leq |r_i|, \forall i \notin S$.

Once we have S, the LMS problem could be solved by finding the maximum error $z = |r_k|$ using the following linear program

$$\begin{aligned}
&\underset{\mathbf{w}}{\text{Minimize}} \quad z && (2)\\
&\text{s.t.} \quad z \geq \left(\sum_{j=1}^{n} w_j x_{ij}\right) - y_i, \quad z \geq y_i - \left(\sum_{j=1}^{n} w_j x_{ij}\right), \quad i = 1, \ldots, m,\\
&\qquad \sum_{j=1}^{n} w_j = 1, \qquad w_j \geq 0, \qquad j = 1, \ldots, n,\\
&\qquad z \geq 0.
\end{aligned}$$

This requires only $n + 1$ decision variables and $2m + 1$ linear constraints if all decision variables are assumed to be positive. The LTS is solved merely by solving Eq. (1) on the given subset. We first describe our experimental setup before testing multiple approaches.

3.1 Random Test Data

We considered two simple data creation methods, differing in the outliers generated in order to detect whether certain LMS or LTS approaches are more susceptible to their structure and distribution throughout the data.

We generated random 5-dimensional vectors such that one or two of the variables held most of the importance (to ensure the potential for high residuals). A random integer q between 400 and 600 was selected for each test, then each w_j calculated as $w_j = j^{q/100} - (j-1)^{q/100}$ before being normalized so that the vectors added to 1. The non-outlier data were generated with $\mathbf{x}_i$ drawn randomly from the unit hypercube (with uniform probability) and y-values calculated using $y_i = \sum\limits_{j=1}^{n} w_j x_j$. Guassian noise was then added with standard deviation $\sigma = 0.05$. Outliers were generated according to two methods. The first method assumes these values are just extra noisy values that follow the same model. Increasing the number of outliers present would not be expected to have a drastic impact on the fitted weighting vectors. The second method strategically positions the values so that the importance of the highest weight should be brought down and the fitted weighting vector would not represent the non-outlier data very well (See Fig. 2).

rand.data.1. $\mathbf{x}$ and y values were determined in the same way as for the non-outlier data, however with $\sigma = 0.1$ and an extra 0.3 added to the y value in the same direction as the noise, i.e. these data points are at least 6 standard deviations (with respect to the noise of non-outlying values) away from values calculated using the model $\mathbf{w}$. Values outside the unit interval were discarded and redrawn.

rand.data.2. $\mathbf{x}$ values are centered according to the generating weighting vector $\mathbf{w}$ with the weights squared and divided by the maximum w_j^2 before Gaussian noise is added with $\sigma = 0.005$. The y values are set to 0 with Guassian noise added $\sigma = 0.1$ and negative values made positive.

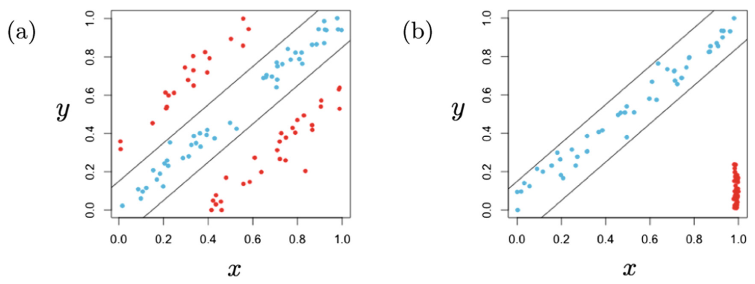

Fig. 2. Structure of random data generated for experiments using (a) rand.data.1 - data randomly distributed at least 6 standard deviations away from the generating function points, and (b) rand.data.2 - data distributed near $y = 0$, close to the corner of the hypercube corresponding to the dimension allocated the highest importance. This data is for the special case of 1 dimension - in our experiments we used 5-dimensional $\mathbf{x}$ vectors. Lines indicate 3 standard deviations (with respect to Gaussian noise of non-outlier data) either side of the generating function.

3.2 Algorithms Based on Random Sampling

We first tested 4 approaches based on Rousseeuw's approach [5,7,8] where we randomly sample sets of n inputs and use them to estimate the weights. In each case, we assume an input dataset consisting of m observed n-dimensional $\mathbf{x}$ inputs and the corresponding y values.

LMS1/LTS1. The weighting vector is initialized at $(1/n, 1/n, \ldots, 1/n)$ and objective value at m. For each of Q iterations, n observed instances are sampled and the corresponding matrix is solved[2] to give the hyperplane through those sampled points. *If* all weights are positive, the squared residual values between this hyperplane and all m points is calculated and the objective value determined (median residual for LMS, mean of smallest 50% of residuals for LTS). If this objective is lower than the current best, the weighting vector

[2] Achieved in `R` using solve(), provided the matrix is non-singular. In the event of singular matrices, the particular iteration contributed nothing to the output.

and best objective value are updated. The iteration is skipped if any of the weights are negative. After Q iterations, the current best weighting vector (not necessarily normalized) and square root of the objective are given as output.

LMS2/LTS2. Same setup as for the LMS1/LTS1 approach, however residuals are still calculated for weighting vectors that include negative values. After Q iterations, the residuals are re-calculated and all inputs with values lower than or equal to the median residual are allocated to the inclusion set S. For LMS, The fitting method of Eq. (2) is then used to find the final weighting vector and the corresponding median residual is then calculated. For LTS, the least squares fitting approach is used on S and then the corresponding root mean squared error of the smallest 50% of residuals according to the resulting weighting vector is calculated. In other words, the method of sampling and solving the system of n points is used to make a best guess at S and then exact fitting approaches are used on S.

LMS3/LTS3. As with previous approaches, subsets of n observations are randomly sampled in each iteration, however rather than solving the linear system, a weighting vector is found by optimizing with respect to the n points (which will always result in appropriate weighting vectors). LMS3a, LTS3a optimize with respect to the maximum error, LMS3b, LTS3b optimize with respect to the least squares criterion and LMS3c and LTS3c optimize with respect to the least absolute deviation. For each iteration, the weighting vector that minimizes the LMS or LTS objective is checked and stored if better than the current best. As with LMS2/LTS2, the best performing vector with respect to the objective is then used to establish the subset S and either the maximum error or least squares are minimized for S.

LMS4/LTS4. Same as LMS3a and LTS3b, however drawing $2n$ random observations.

In each of these methods, thousands of iterations can be used to sample the observations and find the best performing weight vector.

Experiments. To observe the effect of increasing iterations and to compare the approaches, these methods were tested for varying values of Q, and varying number of outliers. In each experiment, 100 non-outlier instances were generated along with 99 outliers, then each method was tested for each setting of $Q = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 100, 200, 500, 1000, 2000\}$, incorporating progressively more of the outliers in 10s, i.e. $10, 20, 30, \ldots, 90$ and 99. There were 100 random datasets generated using each of rand.data.1 and rand.data.2.

Influence of Outliers. Firstly, we are interested in the performance for the highest number of iterations in terms of whether the outliers influenced the weighting vector obtained. In each experiment, the outliers were deemed to have affected the output if the maximum error of the non-outlier data calculated was greater than the minimum error for the outliers.

In the case of data generated by rand.data.1, with the exception of one instance for LMS4, only LMS1 and LTS1 resulted in weighting vectors that were influenced by outliers with increasing frequency as more outliers were included. Table 1 shows the proportion of the 100 experiments where this occurred for each setting of the number of outliers.

Table 1. Prop. of fitted **w** influenced by outliers for LMS1 and LTS1 using rand.data.1

Outliers	10	20	30	40	50	60	70	80	90	99
LMS1	0.01	0	0.01	0.02	0.01	0.02	0.08	0.18	0.24	0.31
LTS1	0	0	0.01	0.01	0.02	0.05	0.07	0.16	0.18	0.32

The main reason these methods may be more susceptible to bad fitting is because an iteration is essentially wasted if the sample generates any negative weights.

For the data generated by rand.data.2, the proportion of tests where the methods resulted in weighting vectors affected by outliers was much higher. We show only LMS1/LTS1, LMS2/LTS2 and LMS3a/LTS3b to give an indication of the performance (all LMS3/LTS3 and LMS4/LTS4 results were similar) (Table 2).

Table 2. Prop. of fitted **w** influenced by outliers for LMS1 and LTS1 with rand.data.2

Outliers	10	20	30	40	50	60	70	80	90	99
LMS1	0	0	0	0.01	0.09	0.43	0.82	0.99	1	1
LTS1	0	0	0	0	0.10	0.31	0.76	0.96	0.99	1
LMS2	0	0	1	0	1	0.17	0.81	1	1	1
LTS2	0	0	0	0	0	0.19	0.67	0.96	1	1
LMS3a	0	0	0.02	0.12	0.28	0.58	0.88	0.99	1	1
LTS3b	0.01	0.01	0.05	0.14	0.31	0.60	0.84	0.98	1	1

For these results, it is not easy to determine whether the outliers exert an influence due to a ‘bad fit’ or because the objective is actually minimized by using the outliers. For example, where there were 50 outliers present, LMS1 had 9 instances where the resulting model was influenced by the outliers, however in 4 of those cases an unaffected model with a better objective was achieved using LMS3a. Conversely, of the 28 affected models using LMS3a, for 4 of these, there was an unaffected model with better error using LMS1. We can infer that the error rate in the presence of this many outliers with this structure in the data can be similar for affected and unaffected models.

Influence of Iterations. Our next question is how many iterations are required to achieve a good level of accuracy. For these particular data generation methods, all methods except for LMS1/LTS1 actually achieved a reasonable accuracy once the number of iterations was above 15, with only marginal improvements after 100. This is not overly surprising, since with these datasets and approaches, if the 5 sampled points happen to be non-outliers then the sampling should identify the plane closest to the non-outlying set and the final step should obtain the optimal error measure. The best methods overall for varying number of outliers were those that used LAD on the random subsets. Figure 3 shows the improvement from 5 to 100 iterations for all methods except for LTS1/LMS1 on both datasets with 50 outliers. LTS1 and LMS1 were not comparable to the remaining methods until the number of iterations was above 500 and at 2000 performed worst overall.

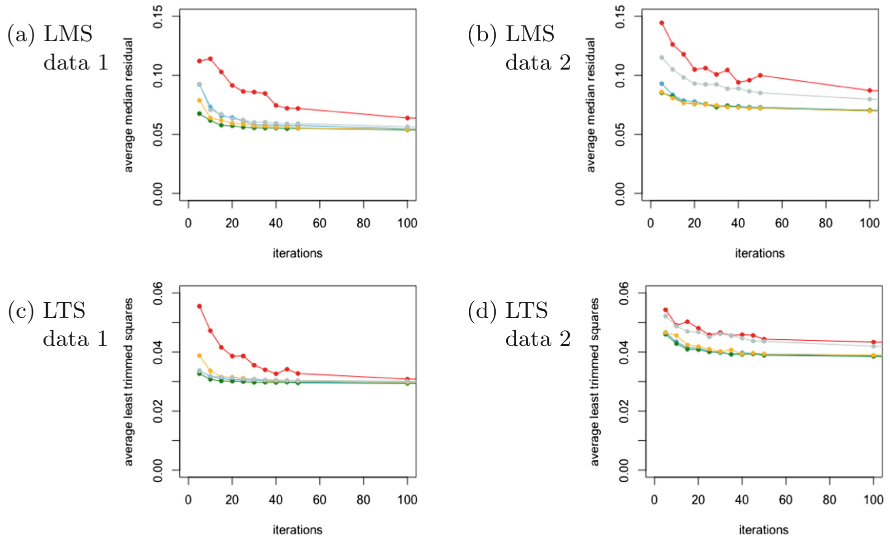

Fig. 3. Average error measures performance over 100 tests with 50 outliers present using rand.data.1 (data 1) and rand.data 2 (data 2). Red = LMS2/LTS2, Blue = LMS3a/LTS3a, Green = LMS3b/LTS3b, Yellow = LMS3c/LTS3b and Grey = LMS4/LTS4. (Color figure online)

Running Time. Lastly we can comment on the time taken to execute the algorithms, which increased close to linearly with the number of iterations. Table 3 shows average times for each of the methods. The LMS3b/LTS3b methods were the slowest, since implementation of LAD requires two extra decision variables for each observation.

3.3 General-Purpose Optimization

We can also look to whether general-purpose solvers can achieve a better trade-off between accuracy and time. We consider two multivariate optimization methods:

Table 3. Time taken on average (in seconds) to run each method with 100 non-outlier and 50 outlier data.

Iterations	LMS						LTS					
	1	2	3a	3b	3c	4	1	2	3a	3b	3c	4
100	0.003	0.047	0.081	0.103	0.065	0.088	0.003	0.049	0.065	0.102	0.081	0.075
500	0.013	0.228	0.392	0.461	0.307	0.419	0.014	0.242	0.308	0.471	0.401	0.317
1000	0.028	0.449	0.765	0.917	0.596	0.821	0.029	0.453	0.602	0.929	0.774	0.617
2000	0.056	0.882	1.523	1.820	1.173	1.625	0.059	0.888	1.198	1.839	1.539	1.231

the derivative based L-BFGS-B method (Broyden-Fletcher-Goldfarb-Shanno with lower and upper box constraints [16]) and the derivative-free COBYLA method (Powell's method of constrained optimization by linear approximations [17]).

LMS5/LTS5. L-BFGS-B only allows for box constraints, so we define an objective function that first normalizes the weighting vector and then calculates the median residual or least trimmed squares. The L-BFGS-B method is then used to optimize with respect to this function. Multiple random-start iterations can be used since the result of L-BFGS-B depends on the initial setting for $\mathbf{w}$.

LMS6/LTS6. COBYLA allows for the constraint on the weighting vector to be imposed via two inequality constraints. Multiple random-start iterations can also be employed here.

Experiments. Numerical experiments were conducted with the same setup as for the random sampling techniques. For 100 random test datasets, we compared LMS5/LTS5 and LMS6/LTS6 with LMS3b/LTS3b. The same settings were used for increasing the number of outliers, while for number of random starts we tested $\{1, 3, 5, 10, 20, 50, 100\}$. For the comparison we set the number of iterations to 100 times the number of random starts for the general methods (as this was anticipated to be comparable in terms of time taken).

Influence of Outliers. For rand.data.1, for the highest number of random starts and iterations tested, LMS5 and LMS6 were influenced by outliers only for high number of outliers present. For 90 outliers, 2 and 1 instance respectively were influenced, while for 99 outliers, this rose to 33 and 8 respectively. In fact, even for 20 random starts, it was still only these two methods that were susceptible. For rand.data.2, all methods were similarly susceptible to outliers. For 50 outliers, the LMS methods had between 24 and 26 tests affected by outliers, while for LTS this was 30–31. Where there was 30 outliers, each method only had one instance where the outliers affected the result.

Influence of Number of Random Starts. As the methods were similarly affected by outliers, we can consider the accuracy in terms of median residual and least trimmed squares values obtained with respect to increases in the number of random starts (or iterations for LMS3b/LTS3b). The general optimization methods were more inaccurate where the number of random starts was below 50, however beyond this the methods were comparable.

Running Time. The time taken to run LMS3b/LTS3b was comparable to LMS6 and LTS5, i.e. LMS used with the derivative-free COBYLA method and LTS with L-BFGS-B, however LMS with L-BFGS-B and LTS with COBYLA took considerably longer. This makes some sense given that once the non-outlier data are found, the LTS problem is essentially a smooth quadratic problem while for LMS there are points of discontinuity in the optimization function. Table 4 shows the average results, showing LMS6 and LTS3b to be slightly faster overall over these tests although not significantly.

Table 4. Time taken on average to run general-purpose solvers (LMS5/LTS5, LMS6/LTS6) compared with LMS3b/LTS3b for data generated by rand.data.1 with 10 outliers. Iterations* indicates number of random starts for general-purpose solvers and number of iterations divided by 100 for LMS3b/LTS3b, i.e. 20 represents 2000.

Iterations*	LMS			LTS		
	3b	5	6	3b	5	6
1	0.087	0.428	0.073	0.084	0.093	0.256
3	0.250	1.399	0.229	0.252	0.266	0.774
5	0.416	2.289	0.380	0.418	0.460	1.371
10	0.825	4.542	0.771	0.838	0.912	2.630
20	1.651	9.145	1.547	1.666	1.808	5.249
50	4.124	22.436	3.805	4.170	4.563	13.408
100	8.245	45.162	7.649	8.350	9.049	26.902

4 Conclusions and Future Work

We have tested various approaches to LMS and LTS fitting of the weighted arithmetic mean. Overall we found that random sampling techniques were fairly competitive with general purpose solvers, however in the latter case there could be improvements made by fine-tuning some of the parameters or altering the objective functions slightly to make them smoother. We did investigate peeling methods, i.e. removing outer points based on the initial optimization, however these were not competitive for the techniques we have shown results for.

We can recommend the LMS3b or LMS6 approaches for fitting to the median residual and LTS3b or LTS5 for fitting with respect to least trimmed squares, although there is much more to investigate.

The techniques could be extended to other aggregation functions with some additional problems arising in some cases, e.g., in the case of the Choquet integral, a random sample of observations, even if 2^n are taken, would not necessarily cover all orderings and hence all simplexes over which the Choquet integral needs to be defined. It also has many additional constraints.

References

1. Beliakov, G., Pradera, A., Calvo, T.: Aggregation Functions: A Guide for Practitioners. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73721-6
2. Beliakov, G.: Construction of aggregation functions from data using linear programming. Fuzzy Sets Syst. **160**, 65–75 (2009)
3. Beliakov, G., Bustince, H., Calvo, T.: A Practical Guide to Averaging Functions. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-24753-3
4. Gagolewski, M.: Data Fusion: Theory, Methods and Applications. Institute of Computer Science, Polish Academy of Sciences, Warsaw (2015)
5. Rousseeuw, P.J.: Least median of squares regression. J. Am. Stat. Assoc. **79**(388), 871–880 (1984)
6. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2017)
7. Dallal, G.E., Rousseeuw, P.J.: LMSMVE: a program for least median of squares regression and robust distances. Comput. Biomed. Res. **25**, 384–391 (1992)
8. Rousseeuw, P.J., Hubert, M.: Recent developments in PROGRESS. In: L_1-Statistical Procedures and Related Topics. IMS Lecture Notes - Monograph Series, vol. 31, pp. 201–214 (1997)
9. Farebrother, R.W.: The least median of squared residuals procedure. In: Farebrother, R.W. (ed.) L_1-Norm and L_∞-Norm Estimation. BRIEFSSTATIST, pp. 37–41. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36300-9_6
10. Grabisch, M., Marichal, J.L., Mesiar, R., Pap, E.: Aggregation Functions. Cambridge University Press, Cambridge (2009)
11. Torra, V., Narukawa, Y.: Modeling Decisions: Information Fusion and Aggregation Operators. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-68791-7
12. Beliakov, G., James, S.: Citation-based journal ranks: the use of fuzzy measures. Fuzzy Sets Syst. **167**, 101–119 (2011)
13. Beliakov, G., James, S.: Using linear programming for weights identification of generalized Bonferroni means in R. In: Torra, V., Narukawa, Y., López, B., Villaret, M. (eds.) MDAI 2012. LNCS (LNAI), vol. 7647, pp. 35–44. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34620-0_5
14. Bloomfield, P., Steiger, W.: Least Absolute Deviations: Theory Applications and Algorithms. Birkhauser, Basel (1983)
15. Bartoszuk, M., Beliakov, G., Gagolewski, M., James, S.: Fitting aggregation functions to data: part I - linearization and regularization. In: Carvalho, J.P., Lesot, M.-J., Kaymak, U., Vieira, S., Bouchon-Meunier, B., Yager, R.R. (eds.) IPMU 2016. CCIS, vol. 611, pp. 767–779. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40581-0_62
16. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**, 1190–1208 (1995)
17. Powell, M.J.D.: A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Gomez, S., Hennart, J.P. (eds.) Advances in Optimization and Numerical Analysis. MAIA, vol. 275, pp. 51–67. Kluwer Academic, Dordrecht (1994). https://doi.org/10.1007/978-94-015-8330-5_4