Computing L1 Straight-Line Fits to Data

Ian Barrodale[1] *(ianbarrodale@gmail.com)*

An L1 Algorithmic and Coding Challenge

In 1971 Frank D.K. Roberts and I developed the *BR algorithm* ([i],[ii]), complete with Fortran code [iii], for calculating L1 solutions to overdetermined systems of m linear equations in n unknown parameters[2]. The challenge here is to develop fast code for L1 fitting by straight lines (n = 2) to big data sets ($m \geq 10^6$) that largely avoids the frustration of sluggish run times. A 25-page report is provided[3] to assist in implementing a *BR Line Fitting (BRLF) algorithm.* Its introduction is intended for those not familiar with L1 approximation, whereas the remainder of the report should also interest readers already acquainted with the inner workings of L1 algorithms. Modern analytics services can provide real-time analysis of streaming data on a continuous basis, and code for a fast BRLF algorithm could be a useful addition for certain applications (e.g., [iv]). The unprecedented growth in number, size and type of real data sets (including sensor data from industrial equipment, cars, aircraft, smart meters, wearable and medical devices, satellites, plus data from social media sites, browser logs, smart phones, financial transactions, gene sequencing) will fuel further innovative applications serving multiple aspects of our lives. This data explosion also serves to provide large and diverse testbeds for algorithm development and evaluation.

L1 refers to a norm defined by *sums of absolute differences* between given data and an approximation to that data. In the simplest case of fitting a straight line to three distinct data points, a best L1 line must pass through (*interpolate*) the two end points. But if the data set contains four distinct points, there may be infinitely many best L1 straight-line approximations, as with the data set {(1,0), (2,1), (3,1), (4,0)}. Since it is known that for any data set a best L1 straight line exists that interpolates at least two of its points, with a four-point data set we need only choose between six lines at most. This is not a practical technique when fitting straight lines to larger data sets[4], since computational experience with [iii] shows that usually relatively few pairs of data points need be examined. Even for million-point data sets used in evaluating some of the modifications described in the report, solutions (which were computed much faster than with [iii]) typically involved only about a dozen pairs of candidate interpolation points.

The original BR algorithm is an adaptation of the Simplex method of linear programming. It found some early use supplementing existing methods for data cleaning, and in interpretation and prediction of small to medium size data series. The proposed BR enhancements will significantly improve its execution speed and facilitate L1 straight-line fits to today's much larger data sets than those of five decades ago.

[i] Barrodale, I. and F.D.K. Roberts (1972) "An improved algorithm for discrete $\ell 1$ linear approximation", TSR 1172, MRC, University of Wisconsin-Madison. Recd. August 17, 1971. https://dspace.library.uvic.ca/handle/1828/11491
[ii] Barrodale, I. and F.D.K. Roberts (1973) "An improved algorithm for discrete $\ell 1$ linear approximation", *SIAM J. Numer. Anal.* **10**, 839-848. Recd. September 29, 1971. https://doi.org/10.1137/0710069
[iii] Barrodale, I. and F.D.K. Roberts (1974) "Algorithm 478 – Solution of an overdetermined system of equations in the $\ell 1$ norm [F4]", *Comm. ACM* **17**, 319-320. Recd. August 4, 1972. https://doi.org/10.1145/355616.361024
[iv] Barrodale, I. (1968) "$L1$ approximation and the analysis of data", *Appl. Statist.* **17**, No. 1, 51-57. https://doi.org/10.2307/2985267

[1] Adjunct Professor of Computer Science, University of Victoria, Victoria BC Canada.
[2] Google Scholar currently lists more than 1,250 combined citations to this algorithm [ii] and software [iii].
[3] Freely available from either http://hdl.handle.net/1828/11460 or from http://arxiv.org/abs/2001.00813.
[4] Nevertheless, those with access to a massively parallel computer might like to experiment with this approach.