Discrete Applied Mathematics 7 (1984) 79–91
North-Holland

A NEW LAD CURVE-FITTING ALGORITHM: SLIGHTLY OVERDETERMINED EQUATION SYSTEMS IN L_1

Eugene SENETA
Department of Mathematical Statistics, Sydney University, Sydney, N.S.W. 2006, Australia

W.L. STEIGER*
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, USA

Received 10 August 1982

We present some ideas for a new algorithm for the LAD curve-fitting problem for n points in $k \le n$ dimensional space. When k is about $\frac{1}{3}n$ it begins to outperform the best current methods, and the advantage increases with k. The algorithm should be of interest in approximation theory and in robust regression. Moreover our approach may be useful in linear optimization, expecially in linear programming.

1. Introduction and summary

Given n points $(x_i, y_i) \in R^{k+1}$, the discrete L_1, or least absolute deviation (LAD) curve-fitting problem seeks $c \in R^k$ to minimize the absolute deviation (AD) distance function

$$f(c) = \| y - Xc \|_1 = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{k} x_{ij} c_j \right| \tag{1}$$

where, under obvious notation, we have written $y' = (y_1, \dots, y_n)$ and the ith row of X is x_i. The minimizing c determines a k-dimensional hyperplane

$$\pi = \{(x, y) \in R^{k+1} : y = \langle c, x \rangle\}$$

that 'best' fits the n points in the sense of discrete L_1. In the curve-fitting context, given a function h, n points $t_1, \dots, t_n \in R$, and k basis functions $\phi_1, \dots, \phi_k$, write $y_i = h(t_i)$ and $x_{ij} = \phi_j(t_i)$. The minimizer $\hat{c}$ defines $P_k = \sum_{j=1}^{k} \hat{c}_j \phi_j$, the k-term approximation that best fits h on the t_i's in the LAD sense; that is, $\sum_{i=1}^{n} |h(t_i) - P_k(t_i)|$ is minimal.

The LAD fitting method seems to have greater antiquity than least squares. In 1789 Laplace (see Eisenhart [11]) developed an algorithm to minimize (1) when

* Research supported in part by a visiting fellowship in the Department of Statistics, The Australian National University, and partly while visiting the Department of Statistics, Princeton University. Computations carried out on the Rutgers Laboratory for Computer Science Research DECSYSTEM 20.

0166-218X/84/$3.00 © 1984, Elsevier Science Publishers B.V. (North-Holland)

$k=1$. It was stimulated by a geometric method of Boscovich [11] when

$$\sum_{i=1}^{n} y_i = c_1 + c_2 \sum_{i=1}^{n} x_i;$$

this is the case $k=2$ with $x_{i1}=1$ and the fit constrained to pass through the centroid of the data.

In 1809 Gauss (see Sheynin [19, p. 311]) stated without proof that a solution c to (1) satisfies $y_i = \langle c, x_i \rangle$ for a certain set of k points (x_i, y_i). This fact would imply a crude algorithm that minimizes f by searching all the $\binom{n}{k}$ c's determined by k of the n given points.

Much later Edgeworth [9, 10] formulated a general procedure for (1) and in 1930 Rhodes [15] devised a method for $k=3$ that seems easy to generalize, as pointed out by Singleton [20]. None of these methods was implemented when computers began to appear, possibly because they were too complicated. In fact it may be the computational complexity of LAD (as well as the analytical intractability in a statistical setting) that forced it to take the back seat to least squares in linear regression.

The connection with linear programming (LP) [7, 12, 23] and the development of Simplex codes led to the first automatic procedures for LAD curve fits. Subsequently there have been many contributions to the subject, among them [1, 8, 13, 14, 17, and 22]. In addition the 1977 Communications in Statistics, Vol. B6, No. 4, is devoted to LAD and contains a comprehensive reference list. See also [25].

At least three good LAD algorithms have been implemented. Barrodale and Roberts (BR) [3, 4] brought about an important advance by showing how to implement an LP version of (1) in an efficient way by modifying the Simplex method. Later, Bartels, Conn and Sinclair (BCS) [5] used a projected gradient method to derive what turns out to be a very similar algorithm. Finally Bloomfield and Steiger (BS) [6] have proposed yet another method that may reduce the complexity of minimizing (1) even further. All three fit into the general framework laid out by Robers and ben Israel [16], BR even matching up in the detailed computations.

Anderson and Steiger [2] discuss the similarities and differences between the three methods and study their computational complexities. It is argued that BCS and BS are linear in n and that all three are linear in k; as n increases, the complexity of BR seemed to increase faster than linearly, probably due to an inefficient line search.

In the present paper we address the problem of limiting the computational cost as k increases towards n. This would be of interest in the curve fitting context, in robust regression and in exploratory data analysis. We convert (1) into an equivalent problem, one that becomes easier to solve as k grows. Specifically, given y and X, the problem of minimizing f in (1) may be written as

$$\min \|r\|_1 : \quad r = y - Xc, \quad c \in R^k. \tag{2}$$

Suppose X has p independent columns, $p \le k$. If A is an $n-p$ by n matrix whose rows are orthogonal to the span of the columns of X (i.e., AX an $(n-p)\times k$ zero matrix), then $Ar=Ay$. Also define $b=Ay\in R^{n-p}$. If the rows of A are also linearly independent, (2) is actually equivalent to

$$\min g(r)\colon \quad Ar=b, \quad \text{where } g(r)=\|r\|_1. \tag{3}$$

This follows from the fact that for any $c\in R^k$, if $r=y-Xc$, $Ar=Ay$, while if $r\in R^n$ satisfies $A(r-y)=0$, $r-y$ is in the (p-dimensional) orthogonal complement of the row space of A (so $r=y-Xc$ for some $c\in R^k$).

The correspondence provides further information that is relevant to iterative algorithms for (2) and (3). As Gauss observed, an r that minimizes (2) may be chosen so that at least k of its components are zero. The same property also characterizes solutions to (3).

Algorithms for (1) and (2) exploit the characterization and move to the optimum via a finite sequence $r_1,, r_2, \ldots, r_N$ of approximations for which $g(r_s)\geq g(r_{s+1})$ and at least k components of r_s are zero. Accordingly, if the jth component of r_N is zero for $j\in B=\{j_1,\ldots,j_k\}$, then c that minimizes f is the solution of the system

$$\sum_{j=1}^{k} c_j x_{ij}=y_i, \quad i\in B.$$

The non-zero components of the optimal r satisfy

$$r_i=y_i-\sum_{j=1}^{k} c_j x_{ij}, \quad i\notin B.$$

However if c has not been computed, these r_i's are also the solutions to

$$\sum_{j\notin B} a_{ij} r_j=b_i.$$

The problem (3) is of interest in its own right. Seneta [18] points out that it was introduced in a statistical setting by Cauchy.

It is worth noting that (3) is not dual to (1), at least in the sense used in linear programming. It would rather seem that (3) is a problem that is complementary to (1) in the sense that its data, A, are in the orthogonal complement of the original data, X.

Writing the LAD problem in parametric form (1), requires the $n\times(k+1)$ augmented matrix $(X\,|\,y)$. If instead it is written in non-parametric form, (3), the augmented matrix $(A\,|\,b)$ of size $n-p\times n+1$ is required. Assuming $p=k$, the latter is smaller when $k\geq\frac{1}{2}n$.

One might therefore expect a good algorithm for (3) to out-perform the best procedures for (1) as $k\geq\frac{1}{2}n$ increases, even taking account of the cost in finding A and b. This turns out to be the case but surprisingly, the cross over point seems to be about $k=\frac{1}{3}n$.

Our algorithm for (3) is based on BS but other LAD procedures could be adapted in a similar way. The idea of minimizing (1) via (3) seems to carry over to functions other than the discrete L_1 norm. For example algorithms for

$$\operatorname{opt} h(r)\colon \quad r=y-Xc, \quad c\in R^k, \quad h: R^n\to R \text{ nonlinear} \tag{4}$$

might be simplified using the foregoing ideas, but we do not address this issue here.

Finally (1) via (3) has bearing on bounded, feasible LP problems. The problem

$$\max \langle d, x\rangle \qquad \text{subject to } \begin{cases} Cx = a, \\ x \geq 0 \end{cases} \tag{5}$$

where C is m by n, $m<n$, is equivalent (see [5] or [21]) to a LAD fit for $n+1$ points in R^{m+1}. In non-parametric form, (3), it would use a data structure of size $n-m+1$ by $n+2$, assuming the row rank of C is m, and is smaller than (5) when $m \geq (n+1)^2/(2n+3)$, or about $\frac{1}{2}n$. If C is dense, the advantage in data reduction becomes apparent.

In Section 2 we give the details of the algorithm for (3), including the passage from (1) to (3). Section 3 contains the results of computational experiments that compare the performance of the new algorithm to BS.

2. The new algorithm

Suppose $A' = (A \mid b)$ is given, A an m by $n>m$ matrix of full rank and $b \in R^m$. The algorithm we give for (3) is, except for some details, quite similar to BS.

As an initial solution take $r = \binom{z}{0}$, $0 \in R^{n-m}$ the zero vector, and write $A = (B \mid N)$, B the $m \times m$ matrix consisting of the first m columns of A, assumed without loss of generality to be linearly independent, N the remaining $n-m$ columns. B is the initial *basis* and corresponds to the m (generally) non-zero elements of r. Because $Ar = b$, $z = B^{-1}b$.

To continue, a column of B, say the pth, will leave the basis via an exchange with a column of N, say the qth, that will enter: z_p will become 0 and 0_q will become $t \neq 0$. Our method will select q in a heuristic fashion and, once it is determined, p will be optimally chosen.

Assuming that n_q has been chosen to enter, the next approximation will be found as a member of the one-parameter family

$$r'(t) = \begin{pmatrix} z'(t) \\ 0 \end{pmatrix} + te_{m+q}$$

where $e_i \in R^n$ is the ith unit coordinate vector and $z'(0) = z$. Clearly $r'(0)$ is the current solution.

Since $Ar'(t) = b$, $Bz'(t) + tn_q = b$, n_q denoting the qth column of N. Thus

$$z'(t) = B^{-1}b - tB^{-1}n_q = z - tv \tag{6}$$

where we write $v = B^{-1}n_q$. The current value of the objective g in (3) will now be

$$g(r'(t)) = \sum_{i=1}^{n} |r_i'(t)| = \sum_{i=1}^{m} |z_i'(t)| + |t|$$
$$= \sum_{i=1}^{m} |z_i - tv_i| + |t|. \tag{7}$$

The value of t that minimizes (7) defines a line $y=tx$ that is the LAD fit through the origin for the points (v_i, z_i), $i=1,\dots,m$ and $(1,0)$. It is easily recognized to be the weighted median $\hat{t}$ of the ratios z_i/v_i, with weight $|v_i|$, and 0, with weight 1 (see [2] or [6]).

If $\hat{t}=0$, $\boldsymbol{n}_q$ may not enter the basis, so another column of N would be investigated. Otherwise $\hat{t}=z_p/v_p$ for some p, $1\le p\le m$, and the pth term in the sum in (7) becomes zero. This means that $z_p'(\hat{t})=0$ so that $\boldsymbol{n}_q$ replaces $\boldsymbol{b}_p$ (the pth column of B) in the next basis. The objective in (7) cannot increase because when $t=0$, $\boldsymbol{r}'(t)=\boldsymbol{r}$ is the current solution and $g(\boldsymbol{r}'(0))\ge g((\boldsymbol{r}'(\hat{t}))$.

We now describe the choice of $\boldsymbol{n}_q$, the non-basic column that will enter. Ideally one seeks that column which, when it optimally replaces some basis column, will produce the greatest reduction in the objective (3). A brute force 'look ahead' method would compute $v=B^{-1}\boldsymbol{n}_q$ for each column $\boldsymbol{n}_q$ of N, minimize (7), and then enter that column corresponding to the smallest value of $g(\boldsymbol{r}'(\hat{t}))$. Our heuristic method, based on that of BS, avoids the computational cost of actually minimizing (7) for each $\boldsymbol{n}_q$, and usually chooses the best column anyway.

The heuristic is based on weighted medians. If $\boldsymbol{n}_q$ were chosen, the foregoing procedure for optimally selecting a column to leave the basis would compute $v=B^{-1}\boldsymbol{n}_q$ and minimize (7). The minimum occurs at $\hat{t}$, the weighted median of the ratios z_i/v_i with weights $|v_i|$, and 0 with weight 1. We assume $\hat{t}\ne 0$ or else $\boldsymbol{n}_q$ cannot enter. One can also express $\hat{t}$ as the median of the distribution function

$$F(t)=\left(I_0(t)+\sum_A |v_i|\right)\Big/\left(1+\sum_{i=1}^m |v_i|\right), \tag{8}$$

where $I_0(t)$ is 1 on $[0,\infty)$ and 0 otherwise, and $A=\{i: z_i/v_i\le t\}$. By definition, $F(\hat{t})\ge\frac12$ and $F(t)<\frac12$ for $t>\hat{t}$.

In (6) the current approximation corresponds to $t=0$, and the next approximation to $\hat{t}$. For this reason the quantity

$$|\tfrac12 - F(0)|, \tag{9}$$

approximately $|F(\hat{t})-F(0)|$, is a rough measure of the distance – in terms of F – between the current approximation and the next one. It attempts to capture the relative advantage of using $\boldsymbol{n}_q$ to optimally replace a basic column.

If in (8) $F(0)=\frac12$, replacement with $\boldsymbol{n}_q$ would not decrease g in (7). Excluding this case, either $F(0)<\frac12$ or $F(t)\to u<\frac12$ as $t\uparrow 0$, by the right continuity of F. The criterion in (9) would then become

$$\begin{array}{ll} \tfrac12 - F(0) & \text{if } F(0)<\tfrac12, \\ F(0^-)-\tfrac12 & \text{if } F(0^-)>\tfrac12. \end{array} \tag{10}$$

From (8),

$$\tfrac12 - F(0)=\tfrac12-\left[\left(1+\sum_{\mathcal{N}}|v_i|+\sum_{\mathcal{P}}|v_i|\right]\Big/\left(1+\sum_{i=1}^m|v_i|\right)$$

which simplifies to

$$\frac{-1-\sum_{\mathcal{N}}|v_i|-\sum_{\mathcal{Z}}|v_i|+\sum_{\mathcal{P}}|v_i|}{2(1+\sum_{i=1}^{m}|v_i|)},$$

where $\mathcal{N}$ denotes the indices for which $z_i/v_i<0$, $\mathcal{Z}$ the indices for which $z_i=0$, and $\mathcal{P}$, those for which $z_i/v_i>0$.

Similarly $F(0^-)-\frac{1}{2}$ may be written as

$$\frac{-1+\sum_{\mathcal{N}}|v_i|-\sum_{\mathcal{Z}}|v_i|-\sum_{\mathcal{P}}|v_i|}{2(1+\sum_{i=1}^{m}|v_i|)}, \tag{12}$$

so the criterion in (10) may be expressed as

$$c_q=\frac{|\sum_{\mathcal{N}}|v_i|-\sum_{\mathcal{P}}|v_i||-1-\sum_{\mathcal{Z}}|v_i|}{2(1+\sum_{i=1}^{m}|v_i|)}. \tag{13}$$

It would be evaluated for each column $\boldsymbol{n}_q$ of N. The one with the largest value of $c_q>0$ is chosen to enter.

If $c_q\le 0$ for all columns in N, the algorithm terminates at an optimal $\boldsymbol{r}$, unless the current approximation is degenerate and $r_j=0$ for $j>n-m$ residuals (so one of the $z_i=0$). This may be seen by interpreting the numerator of (13) as a directional derivative in the direction from $\boldsymbol{r}'$ to $\boldsymbol{r}$. Minus the right hand derivative of (7) at $t=0$ is the numerator of the expression in (11) which therefore measures the rate of decrease in g, as one moves away from the current approximation in the direction

$$\boldsymbol{w}_q=\boldsymbol{r}'(1)-\boldsymbol{r}'(0)=\begin{pmatrix} z-v \\ \boldsymbol{0}\end{pmatrix}+\boldsymbol{e}_{m+q}-\begin{pmatrix} z \\ \boldsymbol{0}\end{pmatrix}=\begin{pmatrix} -v \\ \boldsymbol{0}\end{pmatrix}+\boldsymbol{e}_{m+q},$$

where $v=B^{-1}\boldsymbol{n}_q$. Similarly the left hand derivative at $t=0$ in (7) is the numerator of (12), so it measures the rate of decrease as one moves from the current approximation in the direction $-\boldsymbol{w}_q$.

If $c_q\le 0$, for a certain q, the directional derivative of g along the line $\boldsymbol{r}+t\boldsymbol{w}_q\in\Pi=\{\boldsymbol{x}:A\boldsymbol{x}=\boldsymbol{b}\}$ is always non-negative, by convexity, and $g(\boldsymbol{r})\le g(\boldsymbol{r}+t\boldsymbol{w}_q)$ for all t. If $c_q\le 0$ for all $q=1,\dots,n-m$, then $g(\boldsymbol{r})\le g(\boldsymbol{r}+t\boldsymbol{w}_q)$ for all the $\boldsymbol{w}_q$, so no non-basic column may replace a basic one and reduce g.

If no basic residual is zero, the current $\boldsymbol{r}$ is optimal, by the geometry of the problem. The graph of g is a convex polytope in R^{n+1} comprised of $(n-m)$-dimensional hyperplane 'pieces'. The vertices correspond to points $(\boldsymbol{r},g(\boldsymbol{r}))$ for which $A\boldsymbol{r}=\boldsymbol{b}$ (an $(n-m)$-dimensional hyperplane in R^n) intersected with $n-m$ hyperplanes $r_{i_1}=0$, $r_{i_2}=0,\dots,r_{i_{n-m}}=0$, for any choice of $n-m$ different indices from the set $\{1,\dots,n\}$. Such $\boldsymbol{r}\in R^n$ are 0-dimensional hyperplanes, and there are $\binom{n}{m}$ of them.

Certain pairs of vertices are connected by edges, 1-dimensional hyperplanes in the polytope defined by $\{\boldsymbol{r},g(\boldsymbol{r}):A\boldsymbol{r}=\boldsymbol{b}$ and $r_{i_1}=0,\dots,r_{i_{n-m-1}}=0\}$. A given vertex would have at least $n-m$ edges emanating from it, one corresponding to the relaxation of each of the conditions $r_{i_1}=0,\dots,r_{i_{n-m}}=0$.

A degenerate vertex $(\boldsymbol{r}, g(\boldsymbol{r}))$ has $j > n-m$ edges, one for each condition $r_i = 0$, $i = i_1, \ldots, i_{n-m}$; $j-n+m$ other r_i's $=0$ whilst maintaining $A\boldsymbol{r} = \boldsymbol{b}$. In the non-degenerate case the condition $c_p \leq 0$, all p, guarantees that there is no improvement along any edge, and by convexity in *any* direction in Π.

While it is easy to diagnose degeneracy, it may be complicated to deal with. For this reason we finesse the issue and leave degeneracy as a separate problem, to be treated on its own.

The criterion, (13), for selecting an entering column is analogous to that from the BS LAD algorithm. Other LAD methods like BR and BCS choose the column based only on directional derivatives. BS seems to be the only method using normalized directional derivatives. This has the nice property of removing scaling differences between the $\boldsymbol{n}_q$. The particular normalization used in (13) may partly account for the superior performance of BS. The direction of descent is chosen with reference to weighted medians (9), a concept that characterizes 1-dimensional LAD fits. The L_2 normalized gradient used by Goldfarb and Reid [24] to choose a descent direction for the Simplex method would probably be less effective than (13), on LAD problems. This conjecture is as yet, untested.

A convenient data structure for implementing this algorithm begins with the augmented matrix $A' = (B \mid N \mid \boldsymbol{b})$ premultiplied by B^{-1}. The matrix

$$D = (I \mid B^{-1}N \mid B^{-1}\boldsymbol{b}) \tag{14}$$

contains the information that the vector $\boldsymbol{\sigma} = (\sigma(1), \ldots, \sigma(m))$ of indices pointing to the basic columns is $(1, \ldots, m)$, that the vector $\boldsymbol{\sigma}^c = (\sigma^c(1), \ldots, \sigma^c(\mathrm{n-m}))$ of indices pointing to non-basic columns is $(m+1, \ldots, n)$, and that $z = B^{-1}\boldsymbol{b}$ gives the values of the non-zero components of $\boldsymbol{r} : r_{\sigma(i)} = z_i$ and $r_{\sigma^c(i)} = 0$.

To find the column of N that will become basic, the criterion c_q in (13) is evaluated for each $q = 1, \ldots, n-m$. The v's are the columns of $B^{-1}N$, from (6).

Once the qth element of $\boldsymbol{\sigma}^c$ (call it j; initially $j = \sigma^c(q) = m+q$) has been selected to enter, and the pth element of $\boldsymbol{\sigma}$ is chosen to leave, (14) is updated by a pivot step using Jordan elimination: the pth row of D is multiplied by a constant so the jth column entry is 1; then multiplies of row p are added to each other row so that jth column entries become zero.

If E is the m by m matrix that effects these steps, D now becomes

$$D' = ED = (E \mid EB^{-1}N \mid EB^{-1}\boldsymbol{b}). \tag{15}$$

The new basis pointers are $j' = (1, \ldots, p-1, m+q, p+1, \ldots, m)$ and the non-basic ones are $(\boldsymbol{\sigma}')^c = (m+1, \ldots, m+q-1, p, m+q+1, \ldots, n)$. EB^{-1} is the inverse of the new basis matrix $B' = A(\boldsymbol{\sigma}')$, formed by taking columns from A according to $\boldsymbol{\sigma}'$, and $z' = EB^{-1}\boldsymbol{b}$ are the non-zero components of the new solution $\boldsymbol{r}'$. In general, after several steps $\boldsymbol{\sigma}$ points to m columns of the matrix that comprise an identity, $\boldsymbol{\sigma}^c$ to the $n-m$ columns corresponding to $B^{-1}N$ for the current basis/nonbasis partition of A, and the $(n+1)$st column gives the values of the basic residuals: $r_{\sigma(i)} = z_i$.

Suppose we are given X and y in (1), X an n by k matrix of full rank. To convert the LAD fitting problem into (3) we require an $n-k$ by n matrix A of full row rank that satisfies $AX=0$. Write $A=(B\,|\,N)$, B, $n-k$ by $n-k$, and $X=\binom{X_B}{X_N}$, where X_B is $n-k$ by k and X_N is k by k, and we suppose without loss of generality that X_N is invertible. Then $BX_B+NX_N=0$ and if we take $B=I$, $NX_N=-X_B$. Transposing, N is defined by

$$X_N^{\mathrm{T}}N^{\mathrm{T}}=-X_B^{\mathrm{T}}, \tag{16}$$

$n-k$ linear systems each of size k. Using Gaussian elimination, and backsolving, N can be found in about $\frac{1}{3}k^3+(n-k)k^2=nk^2-\frac{2}{3}k^3$ multiplication and division operations. This initializes (14) with $(I\,|\,N\,|\,\boldsymbol{b})$, where $\boldsymbol{b}=A\boldsymbol{y}$. It is interesting that an initial solution using BS directly on (1) would cost nk^2 steps, so translation from (1) to (3) seems to require no computational overhead.

This description appears in succinct algorithmic form in the appendix. The performance of the algorithm is compared with that of BS on variety of LAD problems, (1). The results appear in the next section.

3. Computational experience

In this section we compare the computational cost of solving (1) directly using BS with that of translating (1) to (3), solving (3) using the algorithm (SS) of the previous section, and then obtaining the optimal $\boldsymbol{c}$. Based on simple assumptions, SS should be cheaper, even when $k<\frac{1}{2}n$, a conjecture that is supported by actual experience.

One iteration of BS costs (in terms of multiplications and divisions) $n(k+1)$ operations to update, plus the cost $\mathrm{WM}(n-k)$ of forming $n-k$ ratios and then finding the weighted median. The first k iterations successively pivot new points into the fit and may be regarded as a start up phase, at total cost of $k[n(k+1)+\mathrm{WM}(n-k)]$. If N further iterations are required the total work for BS would be

$$k[(n(k+1)+\mathrm{WM}(n-k)]+N[n(k+1)+\mathrm{WM}(n-k)]+\tfrac{1}{3}k^3+k^2, \tag{17}$$

the last two terms reflecting the extra cost of solving $\boldsymbol{y}=X\boldsymbol{c}$ for $\boldsymbol{c}$, once it is known which k rows of X determine the optimal fit.

To initialize SS for (3) it costs $nk^2-\frac{2}{3}k^3$ steps to find A plus a further $(n-k)k$ to compute $\boldsymbol{b}=A\boldsymbol{y}$. Each iteration then costs $(n+1)(n-k)+\mathrm{WM}(n-k)$, the first term being the cost of updating D in (14). Thus if M iterations of SS are required, the total work would be about

$$nk^2+nk-\tfrac{2}{3}k^3-k^2+M[(n+1)(n-k)+\mathrm{WM}(n-k)]+\tfrac{1}{3}(n-k)^3+(n-k)^2 \tag{18}$$

steps, the last two terms being the extra cost of finding $\boldsymbol{c}$ once the optimal $\boldsymbol{r}$ is known.

The difference between the BS and SS for just the start up phase and the final solve is therefore (17) − (18), with $N=M$ or

$$k\,\mathrm{WM}(n-k)+\tfrac{2}{3}k^3+k^2+\tfrac{1}{3}[k^3-(n-k)^3]+[k^2-(n-k)^2]. \tag{19}$$

Since WM$(n-k)$ is $n-k$ divisions for the ratios, plus the weighted median calculation, (19) is positive even if k is substantially less than $n-k$.

On the other hand if N and M (the non-startup iterations required by BS and SS respectively) are comparable, that phase of SS will be cheaper than the corresponding one for BS when $k \geq \frac{1}{3}n$. Putting these observations together, one expects SS to outperform BS even when $k < \frac{1}{3}n$, and for fixed n, the advantage should increase with k.

To evaluate the validity of the assumptions on which the foregoing argument is based, we performed actual comparisons of BS with SS. First we approximated $f(x) = \sqrt{x}$ on $[0, 1]$ by the LAD polynomial ϕ_{k-1} of degree $k-1$, based on $n+1$ equally spaced points $t_i = i/n$, $i = 0, 1, \ldots, n$. We thus minimized

$$\sum_{i=0}^{n} \left| \sqrt{t_i} - \sum_{j=1}^{k} c_j t_i^{j-1} \right|,$$

a fit of the form (1) with $y_i = \sqrt{t_i}$ and $x_{ij} = t_i^{j-1}$. We used $n = 15$ and $k = 5, 7, 9, 11$.

The results appear in Table 1. The iteration counts for BS include the k start up steps, so they measure $N + k$. These results do not seem to contradict the assumption that M from (18) and N (from (17) are comparable. Execution timings may be misleading because they reflect features of hardware, and peculiarities of coding. Nevertheless the figures in Table 1 do accord with the conjecture that SS is more efficient than BS when $k > \frac{1}{3}n$ and the advantage improves as k increases.

Next we fit regression models of the form

$$Y = c_1 X_1 + \cdots + c_k X_k + U \tag{20}$$

following the procedue of [2]. Thus a sample of size n from (20) was generated as follows. For each $i = 1, \ldots, n$, successive random numbers $x_{i2}, x_{i3}, \ldots, x_{ik}$, u_i were generated. We set $x_{i1} = 1$ and formed $y_i = c_i x_{i1} + \cdots + c_k x_{ik} + u_i$. The c_j were taken to be $\sqrt{j}$ so the columns of $X = (x_{ij})$ would be scaled differently.

The x_{ij} and u_i were taken to be centered Pareto random numbers of index $\alpha = 1.2$, generated using the density function

$$P(t) = \alpha / [1 + (t-a)^{1+\alpha}], \quad t \geq a = -\alpha/(\alpha - 1).$$

They would have mean zero and infinite variance.

Table 1
Comparison of CPU times and iteration counts for LAD approximation of $\sqrt{x}$ on $[0, 1]$ by $\sum_{j=1}^{k} c_j t^{j-1}$ based on 16 equally spaced points $t_i = i/15$, $i = 0, \ldots, 15$

	$k=5$		$k=7$		$k=9$		$k=11$	
	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
BS	0.036	11	0.044	12	0.082	16	0.088	12
SS	0.032	7	0.042	7	0.046	9	0.042	6

Table 2
Comparison of CPU times and iteration counts summed over 10 independent repetitions of fitting the model (20) for $n=10$ and X, U distributed as Pareto (1.2)

	$k=4$		$k=6$		$k=8$	
	CPU	ITER	CPU	ITER	CPU	ITER
BS	0.16	65	0.28	86	0.32	94
SS	0.18	31	0.16	25	0.12	10

Table 3
Comparison of CPU times and iteration counts summed over 10 independent repetitions of fitting the model (20) for $n=50$ and X, U distributed as Pareto (1.2)

	$k=18$		$k=22$		$k=26$		$k=30$		$k=34$	
	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
BS	7.46	324	10.04	362	13.06	395	17.20	452	20.70	473
SS	6.50	180	7.16	189	7.34	166	7.92	170	8.46	158

Once $(X \mid y)$ is generated LAD estimates of c using BS and SS are obtained. The process is then repeated independently until 10 samples from (20) have been considered. In this way the Monte-Carlo results won't be too strongly influenced by a short sequence from the random number generator.

Table 2 contains results for $n=10$ and Table 3 for $n=50$. The CPU times are net of generating the samples of (20) and are accumulated over the 10 repetitions; the iteration counts are also summed.

In both tables, the iterations differ by about $10k$. Since BS iterations include the k startup steps, this finding again supports the assumption that M from (18) and N from (17) are comparable, since over 10 repetitions we would expect such a difference between the counts. Also the CPU timings suggest that $\mathrm{SS} < \mathrm{BS}$ for $k > \frac{1}{3}n$ and that the difference, $\mathrm{BS} - \mathrm{SS}$ increases with k.

4. Concluding remarks

We have outlined an algorithm, SS, for LAD fitting. Some computational evidence suggests that it may be an interesting competitor to existing procedures as the dimension k grows towards n, the number of points to be fit. However some provisos are necessary.

A practical version of the algorithm would need to use a different data structure because the tableau form in (14) is prone to numerical instability. Similarly the initialization phase could be implemented more stably than the technique suggested by (16).

Finally, it remains to see whether the advantage SS enjoyed over BS would be maintained in better implementations of both algorithms, and over a larger class of problems. It would also be interesting to compare Simplex on dense examples of (5) to SS operating on the non-parametric equivalent LAD fits. These questions are left for further study.

Acknowledgement

We wish to thank Peter Bloomfield for valuable discussions on the equivalence of (1) and (3), and M.R. Osborne for his general wisdom.

Appendix

Given $(x_i, y_i) \in R^{k+1}$, $i=1,\dots,n$ as in (1), write $m=n-k$ and $X=(x_1,\dots,x_n)^T$. The algorithm SS may be described as follows.

Initialize

(1) Renumbering the rows of $(X \mid y)$ if necessary so that X_N, the bottom k rows of X, is invertible, solve the m linear systems $X_N^T N = -X_B^T$ for N, where X_B denotes the top m rows of X.

(2) $D \leftarrow (A \mid b)$ where $A \leftarrow (I \mid N)$ is m by n and $b = Ay$.

(3) $\sigma \leftarrow (1,\dots,m)$, $\sigma^c \leftarrow (m+1,\dots,n)$.

(4) $r_{\sigma(i)} \leftarrow b_i$, $i=1,\dots,m$, $r_{\sigma^c(i)} \leftarrow 0$, $i=m+1,\dots,n$.

Choose entering column

(5) $\mathscr{Z} \leftarrow \{i, 1 \le i \le m : b_i = 0\}$

```
(6) For $j=1$ thru $k$ DO
        $v_i \leftarrow D_{i\sigma^c(j)}, \quad i=1,\dots,m$
        $\mathscr{N} \leftarrow \{i : b_i, v_i \text{ opposite sign}\}$
        $\mathscr{P} \leftarrow \{1,\dots,m\} \setminus \mathscr{N} \setminus \mathscr{Z}$
        $c_j \leftarrow \dfrac{\left| \sum_{\mathscr{N}} |v_i| - \sum_{\mathscr{P}} |v_i| \right| - 1 - \sum_{\mathscr{Z}} |v_i|}{1 + \sum_i^m |v_i|}$
    END
```

(7) $S \leftarrow \{1,\dots,k\}$

(8) $c_q \leftarrow \max_S c_j$

(9) If $c_q > 0$ go to (11)

(10) If $\prod_{i=1}^m r_{\sigma(i)} = 0$ RETURN 'degenerate', else solve $y_{\sigma^c(i)} = \sum_{j=1}^k x_{\sigma^c(i)\,j} \theta_j$ for θ and STOP.

Find leaving column

```
(11) $v_i \leftarrow D_{i,\sigma^c(q)}$, $i=1,\ldots,m$
(12) $\hat{t} \leftarrow$weighted median of $b_1/v_1,\ldots,b_m/v_m$, 0 with weights $|v_1|,\ldots,|v_m|$, 1.
(13) If $\hat{t}=b_p/v_p \neq 0$, go to (16)
(14) $S \leftarrow S\setminus\{q\}$; If $S=\emptyset$, go to (10)
(15) Go to (8)
```

Update

```
(16) Divide row $p$ of $D$ by $D_{p\sigma^c(q)}$
(17) For $i\neq p$, (row $i$ of $D$)$\leftarrow$(row $i$ of $D$) – (row $p$ of $D$) $* D_{i\sigma^c(q)}$
(18) $\sigma(p) \leftrightarrow \sigma^c(q)$
(19) $r_{\sigma(i)} \leftarrow b_i$ $i=1,\ldots,m$; $r_{\sigma^c(q)} \leftarrow 0$
(20) Go to (5)
```

References

[1] N.N. Abdelmalek, Linear L_1 approximation for a discrete point set and L_1 solutions of overdetermined equations, J. ACM 18 (1971) 41–47.

[2] D. Anderson and W.L. Steiger, A comparison of methods for discrete L_1 curve fitting, Department of Computer Science Technical Report, No. 96, Rutgers University (1980).

[3] I. Barrodale and F.D.K. Roberts, An improved algorithm for discrete L_1 linear approximation, SIAM J. Numer. Anal. 10 (1973) 839–848.

[4] I. Barrodale and F.D.K. Roberts, Algorithm 478: Solution of an overdetermined system of equations in the L_1 norm, Comm. ACM 14 (1974) 319–320.

[5] R. Bartels, A. Conn and James W. Sinclair, Minimization techniques for piecewise differentiable functions: The L_1 solution to an overdetermined systems, SIAM J. Numer. Anal. 15 (1978) 224–241.

[6] P. Bloomfield and W.L. Steiger, Least abolute deviations curve-fitting, SIAM J. Scientific and Statistical Comp. 1 (1980) 290–301.

[7] A. Charnes, W.W. Cooper and R.O. Ferguson, Optimal estimation of executive compensation by linear programming, Management Sci. 1 (1955) 138–151.

[8] Jon Claerbaut and Francis Muir, Robust modeling with erratic data, Geophysics 38 (1973) 826–844.

[9] F.Y. Edgeworth, A new method of reducing observations relating to several quantities, Phil. Mag. (Fifth Series) 24 (1887) 222–223.

[10] F.Y. Edgeworth, On a new method of reducing observations relating to several quantities, Phil. Mag. (Fifth Series) 25 (1988) 184–191.

[11] C. Eisenhart, Boscovich and the combination of observations, in: L.L. Whyte, ed., Roger Joseph Boscovich (Fordham University Press, New York, 1961).

[12] T. Harris, Regression using minimum absolute deviations, Amer. Statistician 4 (1950) 14–15.

[13] Otto J. Karst, Linear curve fitting using least deviations, J. Amer. Stat. Assoc. 53 (1958) 118–132.

[14] G.F. McCormick and V.A. Sposito, Using the L_2 estimator in L_1 estimation, SIAM J. Numer. Anal. 13 (1976) 337-343.

[15] E.C. Rhodes, Reducing observations by the method of minimum deviations, Phil. Mag. (Seventh Series) 9 (1930) 974-992.

[16] P. Robers and A. ben Israel, A suboptimization method for interval linear programming: A new method for linear programming, Linear Algebra Appl. 3 (1970) 383-405.

[17] E.J. Schlossmacher, An iterative technique for absolute deviations curve fitting, J. Amer. Stat. Assoc. 68 (1973) 857-865.

[18] EE. Seneta, On a contribution of Cauchy to linear regression theory, Ann. de la Société Scientifique de Bruxelles 90 (1976) 229-235.

[19] O.B. Sheynin, R.J. Boscovich's work on probability, Archive for History of Exact Sciences 9 (1973) 306-324.

[20] R.R. Singleton, A method for minimizing the sum of absolute values of deviations, Ann. Math. Stat. 11 (1940) 301-310.

[21] W.L. Steiger, Linear programming via discrete L_1 curve fitting, Dept. of Computer Science Technical Report, No. 97, Rutgers University (1980).

[22] K.H. Usow, On L_1 approximation II: Computation for discrete functions and discretization effects, SIAM J. Numer. Anal. 4 (1967) 233-244.

[23] Harvey M. Wagner, Linear programming techniques for regression analysis, J. Amer. Stat. Assoc. 54 (1959) 206-212.

[24] D. Goldfarb and J.K. Reid, A practicable steepest-edge simplex algorithm, Math. Programming 12 (1977) 361-371.

[25] P. Bloomfield and W.L. Steiger, Least Absolute Deviations: Theory, Applications, and Algorithms (Birkhauser, Boston, MA, 1983).