Fast Regression Quantiles Using A Modification of the Barrodale and Roberts l_1 Algorithm

Murray Fulton

Shankar Subramanian

Richard T. Carson

University of California, Berkeley

February 1985

Fast Regression Quantiles Using A Modification of the Barrodale and Roberts l_1 Algorithm

Keywords: Regression Quantiles, Trimmed Least Squares, l_1 Norm

LANGUAGE

ISO Fortran

DESCRIPTION AND PURPOSE

Purpose

Given a matrix of independent variables, X, and a dependent variable, Y, this algorithm will calculate a specified, Θth, regression quantile in an efficient manner making it possible to estimate regression quantiles for problems with a large number of observations and/or coefficients.

Theory

Koenker and Basset (1978) defined the Θth regression quantile as the solution to the following linear programming problem:

$$\min_{b,c} \sum_{i=1}^{m} \Theta u_i + \sum_{i=1}^{m} (1 - \Theta) v_i \qquad (1)$$

$$Y_i = \sum_{k=1}^{n} (b_k - c_k) X_{ik} + u_i - v_i \qquad (2)$$

for all i, and $b_k, c_k, u_i, v_i \geq 0$. Least absolute deviation (LAD) regression is the important special case where $\Theta = 1/2$.

Applications

The algorithm may be used to estimate the regression quantile estimator proposed by Koenker and Basset (1978). This estimator is used to construct linear regression analogues to L-estimators in the univariate case. It has been used by Koenker and Basset to construct a robust test for heteroscedasticity (Koenker and Basset, 1982a), to calculate the standard errors for least absolute deviation regression coefficients (Koenker and Basset, 1982b), and to estimate empirical quantiles (Basset and Koenker, 1982). The regression quantile estimator is also required as a preliminary estimator when estimating trimmed least squares in the manner proposed by Ruppert and Carroll (1980).

Numerical Method

The algorithm proposed here is a modification of that given by Barrodale and Roberts (1973; 1974) for solving the LAD problem. Because that algorithm is well documented and now widely implemented in statistical packages (SAS [1980], S [Becker and Chambers, 1984]), we will only briefly note those features which enable the large reductions in computation time over the standard linear programming solution to the LAD problem before considering the modifications necessary to solve the more general regression quantile problem.

The Barrodale and Roberts algorithm differs from the standard simplex algorithm in two main ways. First, it divides the problem into two stages. In the first stage, only the b_k and c_k vectors are allowed to enter the basis, thus greatly reducing the number of vectors which must be searched over, especially since the number of observations is generally substantially larger than the number of coefficients to be estimated. This stage ends when n of the b_k or c_k vectors have entered the basis. The second stage achieves similar savings by not allowing any of the b_k or c_k vectors in the basis to leave as the algorithm searches over the u_i and v_i vectors. This stage ends when all of the marginal costs are nonpositive.

The second major difference between the simplex method and the Barrodale and Roberts algorithm is that Barrodale and Roberts realized that in the LAD case the non-negativity constraints on b_k and c_k could be largely ignored since it was possible to switch back and forth between b_k and c_k and u_i and v_i. This allows many intermediate solutions to be bypassed, greatly reducing the number of iterations necessary to solve the problem.

To modify the Barrodale and Roberts algorithm to solve the regression quantile problem, the objective function must be changed to recognize that u_i are now weighted by 2θ while the v_i's have a weight of $2(1 - \theta)$. Note that when $\theta = 1/2$, the problem is reduced to that of minimizing the sum of absolute deviations, $\sum (u_i + v_i)$ with weights on each observation of one. When the $u_i(v_i)$ vector is interchanged with the corresponding $v_i(u_i)$, the sign on the pivot row is changed and the cost associated with the $u_i(v_i)$ vector is replaced with that of the $v_i(u_i)$ vector (i.e., replace θ with $(1 - \theta)$, or vice versa). To do this the correct weights must be attached to the vectors which are in the basis . If Y_i is positive, then u_i will be in the initial basis and the correct weight is 2θ. If Y_i is negative then v_i will be in the initial basis and the correct weight is $2(1 - \theta)$. From that point on the algorithm remains unchanged since the sum of the marginal costs of u_i and v_i remains -2 and the new marginal costs when u_i and v_i are interchanged in the basis can still be calculated by subtracting twice the pivot row from the old marginal costs.

STRUCTURE

SUBROUTINE L1Q(M,N,M2,N2,A,B,TOLER,X,E,S,THETA)

Formal parameters

M	Integer	input: number of equations
N	Integer	input: number of unknowns ($m \leq n$)
M2	Integer	input: set equal to M + 2
N2	Integer	input: set equal to N + 2
A	Real array	input: two dimensional array of size (M2,N2). On entry, the coefficients of the matrix X must be stored in the first M rows and N columns of A
B	Real array	input: one dimensional array of size M. On entry, B must contain the right hand side of the equations
TOLER	Real	input: a small positive tolerance
X	Real array	output: one dimensional array of size N. On exit, this array contains the solution to the regression quantile problem
E	Real array	output: one dimensional array of size M. On exit, this array contains the residuals
S	Integer array	input: array of size M used for workspace
THETA	Real	input: value of theta in the L1Q problem
A(M+1,N+1)	Real	output: minimum sum of the weighted absolute values of the residuals
A(M+1,N+2)	Real	output: rank of the matrix of coefficients
A(M+2,N+1)	Real	output: Exit codes with values 0 - optimal solution which is probably nonunique 1 - unique optimal solution 2 - calculations terminated prematurely due to rounding error
A(M+2,N+2)	Real	output: number of simplex iterations performed

RESTRICTIONS AND TIME

Restrictions

There are no general restrictions except that the number of observations be $\geq$ the rank of the coefficient matrix, with the coefficient matrix full rank. It should be noted, however, that the solution, particularly in data sets where n is not small relative to m, may not necessarily be unique. The Barrodale and Roberts algorithm determines if the solution is unique and returns a code of 1 to indicate this.

Time

The following time comparisons were established on a VAX 11/750 running under the UNIX operating system.

# of Iterations and Computation Time (CPU Seconds) for 4 DATA SETS ($\theta = 0.2$)[1]

	LINDO		MODIFIED BARRODALE AND ROBERTS		OLS
	time	iterations	time	iterations	time
STACK LOSS	8.4	38	8.3	9	8.1
SAVINGS	23.5	78	8.2	13	7.8
AUTO	90.2	195	9.5	22	11.8
BOSTON	na	na	65.3	72	37.2

We used a widely distributed linear programming package, LINDO (Schrage, 1984), to implement the standard lp algorithm for solving for regression quantiles. For both the savings and auto data, LINDO gave the incorrect answer, usually stopping an iteration or two from the correct solution. This clearly illustrates the problem of using standard linear programming to do regression quantiles; the round-off error from numerous iterations can be very serious.

[1] The stack loss regression and data are due to K.A. Brownlee and were used by Ruppert and Carroll (1980). This data set has 21 observations and 4 independent variables including the constant term. The savings data were collected for 50 countries by Arlie Sterling and used as an example by Belsley, Welsch, and Kuh (1980). This data set has 5 independent variables. The auto data is from a study on the characteristics of automobiles; 74 observations on 10 independent variables were used here. The Boston data is from a study of air pollution and housing prices by Harrison and Rubinfeld and was used as an example by Belsley, Welsch and Kuh (1980). This data set has 506 observations and 14 independent variables. All of these data sets are available as part of the S statistical package (Becker and Chambers, 1984).

This problem can be avoided by using a linear programming package such as MPSX (IBM, 1979) which has extended precision features but only at the cost of more computational time. The Boston problem was too large to run in LINDO without modifications to that package. It would have been possible to run this problem using MPSX but only at a prohibitive cost. The OLS times are provided as a benchmark since most readers are familiar with times for OLS calculations on their own systems.

REFERENCES

Barrodale, I. and Roberts, F.D.K. (1973), An Improved Algorithm for Discrete l_1 Linear Approximations. *SIAM Journal of Numerical Analysis*, 10, 839-848.

Barrodale, I. and Roberts, F.D.K. (1974), Algorithm 478: Solution of an Overdetermined System of Equations in the l_1 Norm. *Communications of the Association for Computing Machinery*, 17, 319-320.

Basset, G. and Koenker, R. (1982). An Empirical Quantile Function for Linear Models with iid Errors. *Journal of the American Statistical Association*, 77, 407-415.

Becker, R.A. and J.M. Chambers (1984). *S: An Interactive Environment for Data Analysis and Graphics* (Belmont, CA: Wadsworth).

International Business Machines (1979). *IBM Mathematical Programming System: Extended/370 Program Leference Manual, 4th ed*. (White Plain, NY: IBM).

Koenker, R.W. and Bassett, G.W. (1978). Regression Quantiles. *Econometrica*, 46, 33-55.

Koenker, R.W. and Bassett, G.W. (1982a). Robust Test for Hetroscedasticity Based on Regression Quantiles. *Econometrica*, 50, 43-62.

Koenker, R.W. and Basset, G.W. (1982b). Tests of Linear Hypotheses in the l_1 Norm. *Econometrica*, 50 1577-1583.

Ruppert, D. and Carroll (1980). Trimmed Least Squares Estimation in the Linear Model. *Journal of the American Statistical Association*, 75, 828-838.

SAS Institute (1980). *SAS Supplemental Library Users Guide: 1980 Edition*, (Cary, NC: SAS Institute).

Schrage, L. (1984). *User's Manual: Linear, Integer, and Quadratic Programming with LINDO*, (Palo Alto, CA: Scientific Press).

Sections of Original Barrodale and Roberts LAD Code to Be Modified

```
      subroutine l1(m,n,m2,n2,a,b,toler,x,e,s)

c compute the marginal costs
      do 60 j = 1,n1
      sum = 0.0d0
      do 50 i = 1,m
      sum=sum+a(i,j)
50    continue
      a(m1,j) = sum
60    continue


380   continue
      a(m2,n2) = kount
      a(m1,n2) = n1-kr
      sum = 0.d0
      do 390 i=kl,m
      sum=sum+a(i,n1)
390   continue
```

Changes Necessary for Computing Regression Quantiles

```
      subroutine l1q(m,n,m2,n2,a,b,toler,x,e,s,theta)
      real theta

c compute the marginal costs
      do 60 j = 1,n1
      sum = 0.0d0
      do 50 i = 1,m
      if(b(i).le.0.) go to 45
      sum=sum+2.*theta*a(i,j)
      go to 50
45    sum=sum+2.*(1.-theta)*a(i,j)
50    continue
      a(m1,j) = sum
60    continue


380   continue
      a(m2,n2) = kount
      a(m1,n2) = n1-kr
      sum = 0.d0
c compute weighted sum of residuals
      do 390 i=1,m
      if(e(i).le.0) go to 385
      sum=sum+theta*e(i)
      go to 390
385   sum=sum-(1.-theta)*e(i)
390   continue
```

FAST REGRESSION QUANTILES ALGORITHM

```
c     l1qregression quantiles
      subroutine l1q(m,n,m2,n2,a,b,toler,x,e,s,theta)
c barrodale and roberts, cacm (june 1974) pp 319-320
c algorithm 478
c modified for regression quantiles

      double precision sum
      real min,max,a(m2,n2),x(n),e(m),b(m)
      integer out,s(m)
      logical stage,test
c big must be set equal to any very large real constant
      data big/1.e38/
c initialization
      m1 = m+1
      n1 = n+1
      do 10 j = 1,n
      a(m2,j) = j
      x(j) = 0.
10    continue
      do 40 i = 1,m
      a(i,n2) = n+i
      a(i,n1) = b(i)
      if (b(i).ge.0.) go to 30
      do 20 j = 1,n2
      a(i,j) = -a(i,j)
20    continue
30    e(i) = 0.
40    continue
c compute the marginal costs
      do 60 j = 1,n1
      sum = 0.0d0
      do 50 i = 1,m
      if(b(i).le.0.) go to 45
      sum=sum+2.*theta*a(i,j)
      go to 50
45    sum=sum+2.*(1.-theta)*a(i,j)
50    continue
      a(m1,j) = sum
60    continue
c stage 1
c determine the vector to enter the basis
      stage = .true.
      kount = 0
      kr = 1
      kl = 1
70    max = -1.
      do 80 j = kr,n
      if (abs(a(m2,j)).gt.n) go to 80
      d = abs(a(m1,j))
      if (d.le.max) go to 80
```