Two algorithms for large-scale L1-type estimation in regression

Song Cai

The University of British Columbia

June 3, 2012

Outline

- The optimization problem to be solved
- Statistical background
- A survey of existing methods
- Algorithm I
- Algorithm II
- Numerical results

Problem description

- Mathematical problem:

$$\min_{x \in \mathbb{R}^p} \sum_{i=1}^{K} |a_i x + b_i|,$$

  where a_i's are known p-dimensional vectors and b_i's are known scalars, $i = 1, 2, \cdots, K$. The objective function is convex, but nonsmooth.
- Difficulty: Large scale nonsmooth problems (p – a few hundreds and K – the order of a million) usually demands long CPU time or requires large computer memory.
- Purposes: Design algorithms that are reasonably fast and consume small/moderate amount of memory (the amount that can be found on most modern personal computers)

Statistical background – regression with L1-type loss function

- Consider i.i.d data $\{(x_i, y_i)\}_{i=1}^N$ and linear regression model

$$Y_i = \beta^T X_i + \varepsilon_i,$$

where β is a p-dimensional parameter and ε is a random error.

- A few L1-type estimation methods for model parameter β:
  - Quantile regression (QR) (Koenker and Bassett, 1982) and Least absolute deviation (LAD) estimation
  - Composite quantile regression (CQR) (Zou and Yuan, 2008)
  - Rank estimate with Wilcoxon score (Jaeckel, 1972)

LAD and QR

- Least absolute deviation regression:

$$\hat{\beta}_{LAD} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{N} |y_i - \beta^T x_i|.$$

- Quantile regression: estimate the regression parameter β and the τ's quantile of ε, b_τ, by

$$(\hat{\beta}_{QR_\tau}, \hat{b}_\tau) = \underset{\beta, b}{\operatorname{argmin}} \sum_{i=1}^{N} \rho_\tau(y_i - b - \beta^T x_i),$$

  where $\rho_\tau(t) = \tau t^+ + (1-\tau)t^-$ is called the check function. NOTE: $|t| = t^+ + t^-$ and all algorithms we are going to talk about apply here.

- Dimension of the parameter: p; number of terms to be summed up: $K = N$.

CQR

- Composite quantile regression: fix a positive integer M, and estimate β and the $\tau_1, \tau_2, \cdots, \tau_M$'s quantile of ε, $b_1, b_2, \cdots, b_M$, by

$$(\hat{\beta}_{QR_\tau}, \hat{b}_1, \cdots, \hat{b}_M) = \underset{\beta, b_1, \cdots, b_M}{\operatorname{argmin}} \sum_{j=1}^{M} \left\{ \sum_{i=1}^{N} \rho_\tau(y_i - b_j - \beta^T x_i) \right\}.$$

- Pro: for a large M (say 15), the efficiency of the CQR estimator ($\hat{\beta}_{CQR}$) of β is significantly higher than the QR estimator $\hat{\beta}_{QR}$ of β for a wide range of distributions, e.g. normal, t and Weibull.
- Con: increased computational complexity – dimension of the parameter: p; number of terms to be summed up: $K = M * N$

Rank regression of Jaeckel (1972)

- Rank estimator of β is given by

$$\hat{\beta}_{rank} = \underset{\beta}{\operatorname{argmin}} \sum_{1 \le i < j \le N} |e_i - e_j|,$$

  where $e_i = y_i - \beta^T x_i$, $i = 1, \cdots, N$, is the residual.
- Pros:
  - Highly efficient – the asymptotic variance of $\hat{\beta}_{Rank}$ is equivalent to that of $\hat{\beta}_{CQR}$ when number of quantile used (M) in CQR tends to infinity (a limit that CQR can never reach in practice).
  - Robust to outliers in ε – $\hat{\beta}_{rank}$ has bounded influence in Y space (Chang et al, 1999 JASA).
  - There are high breakdown versions of rank estimator to β which provides further robustness to both outliers in X and ε and also retains the convexity of the objective function, e.g. GR by Naranjo and Hettmansperger (1994, JRSSB) and HBR by Chang et al (1999, JASA)

3

Rank regression of Jaeckel (1972) (cond')

- Being an earlier invented than and theoretically superior method to QR, why rank regression is not as popular as QR/LAD (at least in subject areas)?

- Con: high computational cost: dimension of the parameter: p; number of terms to be summed up: $K = N * (N-1)/2 = O(N^2)$, if sample size $N = 1500$, K is over $1,000,000$.

Rank regression of Jaeckel (1972) (cond')

- Being an earlier invented than and theoretically superior method to QR, why rank regression is not as popular as QR/LAD (at least in subject areas)?

- Con: high computational cost: dimension of the parameter: p; number of terms to be summed up: $K = N * (N-1)/2 = O(N^2)$, if sample size $N = 1500$, K is over $1,000,000$.

Golden oldies

- From late 1960's to mid 1980's, a lot of algorithms have been proposed. To name a few famous ones: BR (Barrodale and Roberts, 1973), AFK (Armstrong, Frome and Kung, 1979), BS (Bloomfield and Steiger, 1980).
- BR (implemented in R quantreg package) has been a golden standard to be compared with because it is reasonably fast using the standard at that time and numerically stable. Most famous algorithms at that time have similar performance to BR.
- For recent development (e.g. extension of BR to constrained optimization) see Shi and Lukas (2002) and references therein.
- Problem: slow (in today's standard) for large scale problem. Try using quantreg with default method "BR" to solve a 1-dimensional optimization problem with $1,000,000$ absolute terms . . .

Golden oldies

- From late 1960's to mid 1980's, a lot of algorithms have been proposed. To name a few famous ones: BR (Barrodale and Roberts, 1973), AFK (Armstrong, Frome and Kung, 1979), BS (Bloomfield and Steiger, 1980).
- BR (implemented in R quantreg package) has been a golden standard to be compared with because it is reasonably fast using the standard at that time and numerically stable. Most famous algorithms at that time have similar performance to BR.
- For recent development (e.g. extension of BR to constrained optimization) see Shi and Lukas (2002) and references therein.
- Problem: slow (in today's standard) for large scale problem. Try using quantreg with default method "BR" to solve a 1-dimensional optimization problem with $1,000,000$ absolute terms ...

Golden oldies (cond') – Code for the test example of slow speed

```
library(quantreg)
beta <- 0.8
x <- rnorm(1000000, 3, 2)
e <- rnorm(1000000, 0, 1)
y <- beta*x + e
rq(y x, method="br")
```

Modern way – use a decent interior-point linear programming solver

- "Correct" way to go today – use a parallel inerior-point linear programming (LP) solver.
- As long as memory (RAM) is big enough, modern interior-point LP solvers provide
  - very accurate result
  - very fast convergence rate (probably the fastest way of solving our optimization problem to a high accuracy level if computer's memory is big enough)
- To name a few fastest parallel commercial LP solvers: Xpress, Mosek, Gurobi, IBM CPLEX

Modern way (cond') – LP formulation of our optimization problem

- Idea: reformulate the nonsmooth problem to a smooth linear optimization problem with linear constraints, but the cost is the much increase number of variables (dimension) and lots of constraints.
- One formulation (there are other formulations, e.g. Li 1998):

$$\min_{u_i} \sum_{i=1}^{K} u_i$$

subject to

$$u_i \geq y_i - \beta^T x_i \text{ for } i = 1, \cdots, K$$

$$u_i \geq -(y_i - \beta^T x_i) \text{ for } i = 1, \cdots, K$$

- Original problem: dimension – p, number of absolute terms – K; New LP problem: dimension – $K + p$, number of constraints – $2K$. When K is $1,000,000$ and p is 100, LP solvers will use HUGE amount of memory.

Modern way (cond') – Code for the test example of very high memory usage

Warning: If your computer has less than 8GB memory, DON'T try. It will crash your computer! If you have 16GB memory, try at your OWN risk, and please let me know your result!

```
library(quantreg)

beta <- rnorm(100, 0, .8)

x <- matrix(rnorm(1000000*100, 3, 2), 1000000, 100)

e <- rnorm(1000000, 0, 1)

y <- x%*%cbind(beta) + e

rq(y x, method="fn")
```

Other algorithms

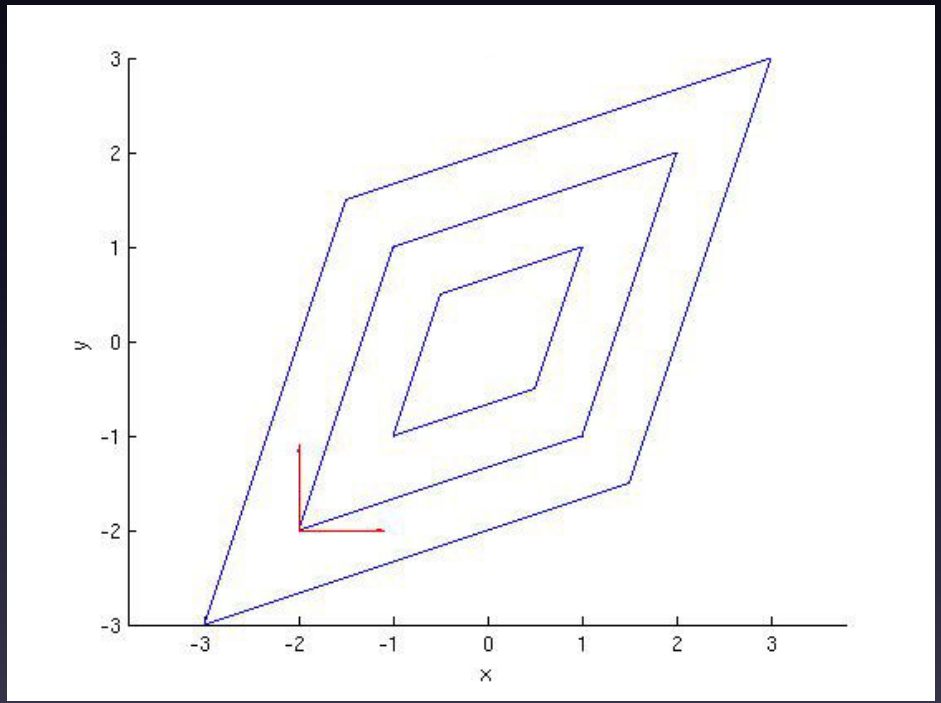
- Subgradient method – slow convergence, no practical stopping rule
- Ellipsoid method – numerical instability for any high dimensional problems (for $p > 10$)
- Coordinate descent algorithm
- More . . .

Mixed coordinate descent and steepest descent algorithm

- Coordinate descent algorithm – a popular algorithm for regularized estimation in regression with convex smooth loss function and LASSO-type penalty. Wu and Lange (2008) have been promoting it for L1-type loss function.
- The algorithm:
  - Start from an initial point.
  - In each iteration step, perform a line search along each coordinate direction and cycle through all coordinates (cyclic version), or perform one line search only along the coordinate direction that brings down the objective value most (greedy version).
  - Iterate until algorithm converges.

Problems with coordinate descent

- Problems of coordinate descent for nonsmooth objective function:
  - May converges to non-optimal kink (no available cures)



  - Convergence may be very slow (cures available, e.g. Loshchilov et al. 2011, Li and Acre 2004 for one-dimensional problem)

Solving the problem of coordinate decent

- Getting stuck at a non-optimal point frequently happens especially for high dimensional problems with sparse data points (the surface of the objective function is very non-smooth).
- One simple idea: if coordinate descent algorithm converges, check for steepest descent direction, if the point found is not global minimum, optimize along the steepest descent direction.
- Searching for steepest descent direction in our case is a quadratic programming problem with simple bounds on variables. And the computational complexity usually is not high.
- A simple example: a problem with $p = 5$, $K = 10$ and minimal objective function value 0. Pure coordinate descent algorithm obtained wrong minimal obj. value of 9.7, while mixed coordinate descent and steepest descent algorithm, with only four additional steepest descent searches, gives 4×10^{-15}.

Modified relaxation method of Camerini, Fratta and Maffioli (CFM), 1975

- Idea: At each step of iteration, find a direction that has smaller angle towards the global minimum than the previous direction.
- Pro: fast convergence (usually faster than coordinate descent and/or steepest descent)
- Cons:
  - Need to know the minimum objective function value (or a good approximation to it) in advance.
  - No formal convergence rule is given in CFM paper, although practical stopping rule for the algorithm is easy to implement
- Our modification provides:
  - better direction search for our particular problem
  - good and improved approximation to the minimum objective value at each iteration
  - a formal convergence (to global minimum) rule (to be implemented)

Numerical experiment

- Set $p = 100$ and $K = 1,000,000$ for our optimization problem, and randomly generate data ten times. For each dataset we try the two proposed algorithms, BR and Mosek interior-point solver and calculate the average times.
- Both our algorithms get correct answer for all datasets.
  - Algorithm 1 (Mixed CD & SD):
    - 330 seconds on average
    - memory used < 1GB
  - Algorithm 2 (Specialized CFM):
    - 37 seconds on average
    - memory used < 1GB
  - BR: took too long to run, terminated manually
  - Mosek interior-point solver: run out of memory (8GB), reset my computer

12

Summary

- When memory is enough, one should use a interior-point solver for high accuracy and fast speed
- For large scale (especially large number of absolute terms K) problem, both proposed algorithms give correct results in reasonable amount of time and use small amount of memory.
- Algorithm 2 (specialized CFM) in general is faster than algorithm 1 (mixed cd and sd), but improvements need to be made to find a formal convergence rule and make better approximation to the minimum objective value.