Regularized Least Absolute Deviations Regression and an Efficient Algorithm for Parameter Tuning

Li Wang

Ross School of Business
University of Michigan, Ann Arbor
wang@umich.edu

Michael D. Gordon

Ross School of Business
University of Michigan, Ann Arbor
mdgordon@umich.edu

Ji Zhu

Department of Statistics
University of Michigan, Ann Arbor
jizhu@umich.edu

Abstract

Linear regression is one of the most important and widely used techniques for data analysis. However, sometimes people are not satisfied with it because of the following two limitations: 1) its results are sensitive to outliers, so when the error terms are not normally distributed, especially when they have heavy-tailed distributions, linear regression often works badly; 2) its estimated coefficients tend to have high variance, although their bias is low. To reduce the influence of outliers, robust regression models were developed. Least absolute deviation (LAD) regression is one of them. LAD minimizes the mean absolute errors, instead of mean squared errors, so its results are more robust. To address the second limitation, shrinkage methods were proposed, which add a penalty on the size of the coefficients. The LASSO is one of these methods and it uses the L1-norm penalty, which not only reduces the prediction error and the variance of estimated coefficients, but also provides an automatic feature selection function. In this paper, we propose the regularized least absolute deviation (RLAD) regression model, which combines the nice features of the LAD and the LASSO together. The RLAD is a regularization method, whose objective function has the form of "loss + penalty." The "loss" is the sum of the absolute deviations and the "penalty" is the L1-norm of the coefficient vector. Furthermore, to facilitate parameter tuning, we develop an efficient algorithm which can solve the entire regularization path in one pass. Simulations with various settings are performed to demonstrate its performance. Finally, we apply the algorithm to solve the image reconstruction problem and find interesting results.

1. Introduction

Linear regression is one of the most important and widely used techniques for data analysis. Let Y be the response variable and $x \in R^p$ be the vector of p independent predictor variables, linear regression assumes the following relationship between Y and x:

$$Y = \beta_0 + x^T \beta + \varepsilon ,$$

where ε is normally distributed. To estimate the coefficients, suppose n samples are obtained. Let $y \in R^n$ be the vector of n observations for the response variable and $X \in R^{nxp}$ be the matrix, where each column corresponds to a predictor and each row corresponds to a sample.

When ε has normal distribution with constant variance and is independent of x, the optimal estimator of β_0 and β can be obtained by minimizing the L_2 norm of the residuals:

$$\begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{pmatrix} = \arg\min(\| y - (\beta_0 + X\beta) \|_2) \tag{1}$$

$(\hat{\beta}_0, \hat{\beta})$ is also called the ordinary least square (OLS) estimator. For simplicity, let $\overline{X} = [\vec{1}, X]$, where $\vec{1}$ is a column vector with all the components being 1. The OLS estimator is calculated from:

$$\begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{pmatrix} = (\overline{X}^T \overline{X})^{-1} \overline{X}^T y \tag{2}$$

A well-known problem with linear regression is that its results are very sensitive to outliers. One way to solve this problem is to identify the outliers via diagnosis and discard them. However, when y does not have normal distribution, e.g., heavy-tailed distribution, simply discarding outliers might not be a good solution. Another way to remedy this problem is to use robust methods. Robust methods minimize other functions of residuals, instead of the L_2 norm function. Least absolute deviation (LAD) regression is one of the robust methods and the LAD estimator is obtained by minimizing the L_1 norm of the residuals:

$$\begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{pmatrix} = \arg\min(\| y - (\beta_0 + X\beta) \|_1) \tag{3}$$

IEEE COMPUTER SOCIETY

Figure 1 shows the loss functions used in linear regression (in dashed line) and LAD method (in solid line). It can be seen from the plot that outliers in response variable have much less influence for LAD. Several studies [2, 3, 5, 7, 9] have confirmed that LAD estimator is more robust than the OLS estimator, especially when the response variable has heavy-tailed distribution and observed values for predictor variables are not contaminated by noise.

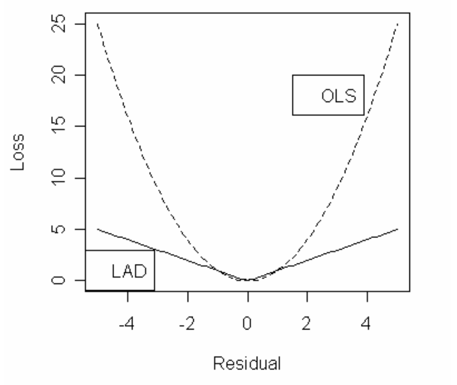


Figure 1. Loss functions for linear regression and least absolute deviation regression

Another problem of linear regression is that its estimated coefficients often have low bias but large variance [10]. From (2), it can be seen that to calculate the OLS estimator the inverse matrix of $\overline{X}^T\overline{X}$ must be solved. Therefore, when $\overline{X}^T\overline{X}$ is ill-conditioned, which happens when some predictors are correlated, the OLS estimator becomes very sensitive to values in *X*. Shrinkage methods are proposed to address this issue. By trading off a little bias, the shrinkage methods can reduce the variance of the estimator and improves the overall prediction accuracy. The LASSO [10] is one of the shrinkage methods and it solves the following optimization problem:

$$\min \| y-(\beta_0+X\beta)\|_2+\lambda\|\beta\|_1 \qquad (4)$$

Note that this is a regularization method, which has the form of *loss* + *penalty*. Imposing the L_1 norm penalty on the coefficient vector has the nice property that some coefficients can be shrunk exactly to 0 when λ is set appropriately. Therefore, an automatic feature selection can be achieved by the LASSO and the selection process is continuous in λ.

In this paper, we combine the LAD and the LASSO together and propose the regularized least absolute deviation (RLAD) regression model, which solves the optimization problem below:

$$\min \| y-(\beta_0+X\beta)\|_1+\lambda\|\beta\|_1 \qquad (5)$$

The RLAD model inherits the nice properties from both methods and its benefits are three-fold: 1) by using the robust loss function, its results are not sensitive to outliers; 2) by using the L_1 norm penalty, both the variance of the RLAD estimator and the prediction error are reduced; 3) it can perform automatic feature selection, where the coefficients for irrelevant features are shrunk to zero.

The regularization parameter λ plays a vital role in regularization methods, since it controls the compromise between *loss* and *penalty*. Therefore, its value must be chosen carefully, because changing λ can often result in estimators with significantly different prediction accuracy. However, there is usually no good way to determine the best value of λ beforehand, so people often need to specify a large number of λ values and test each of them. This parameter tuning process is usually computationally expensive and it does not guarantee that the best λ value can be discovered. Thus, although problem (5) can be solved by calling a linear programming (LP) solver, in practice it is not efficient, because the solver must be invoked extensively during the parameter tuning process. To facilitate parameter tuning, we develop an efficient algorithm, which can solve the entire regularization path in one pass. In other words, the algorithm can give the solutions for every possible value of λ.

The paper is organized as follows. Section 2 describes the algorithm; section 3 uses simulations to demonstrate its performance; section 4 applies the algorithm for solving the image reconstruction problem; section 5 presents the conclusions and describes the future work.

2. The algorithm for RLAD

2.1. Theories for the algorithm

The linear programming problem (5) can be equivalently transformed into:

$$\begin{aligned}
&\min \sum_{i=1}^{n}\varepsilon_i \\
&s.t.\ \ -\varepsilon_i \le y_i-(\beta_0+x_i^T\beta)\le\varepsilon_i,\ \ i=1,..,n \qquad (6)\\
&\qquad \sum_{i=1}^{n} sign(\beta_i)\beta_i \le s\\
&\qquad \varepsilon_i\ge 0,\ \ i=1,..,n
\end{aligned}$$

In this problem, *s* is the regularization parameter (a positive constant) and it controls the size of the coefficient vector. Two problems are equivalent in the sense that for every *s*>0 in (6) we can find a corresponding λ>0 in (5), such that two problems have the same objective value.

Since this is an LP problem and *s* is one of the constraints, from LP theories we know that the optimal solutions are piecewise linear functions of *s*. For readers who are not familiar with LP, this point can also be seen from the following derivations. The basic idea of our algorithm is to continuously increase *s* and solve the solutions on this regularization path, until the objective value can not be further decreased.

The *Lagrangian L* associated with problem (6) is:

$$\begin{aligned} L = &\sum_{i=1}^{n}\varepsilon_i + \sum_{i=1}^{n}\eta_i^+(y_i - (\beta_0 + x_i^T\beta) - \varepsilon_i) - \\ &\sum_{i=1}^{n}\eta_i^-(y_i - (\beta_0 + x_i^T\beta) + \varepsilon_i) + \\ &\lambda(\sum_{i=1}^{n} sign(\beta_i)\beta_i - s) - \sum_{i=1}^{n}\gamma_i\varepsilon_i \end{aligned} \quad (7)$$

In (7), $\eta_i^+, \eta_i^-, \gamma_i$ (*i=1,..,n*) and λ are nonnegative *Lagrangian* multipliers. The Karush-Kuhn-Tucker (KKT) conditions are:

$$\frac{\partial L}{\partial \beta_0} = -\sum_{i=1}^{n}(\eta_i^+ - \eta_i^-) = 0 \quad (8)$$

$$\frac{\partial L}{\partial \beta_j} = -\sum_{i=1}^{n}(\eta_i^+ - \eta_i^-)x_{ij} + \lambda sign(\beta_j) = 0, \quad (9)$$
$$for \quad \beta_j \neq 0$$

$$\frac{\partial L}{\partial \varepsilon_i} = 1 - \eta_i^+ - \eta_i^- - \gamma_i = 0, \quad for \quad i = 1,...,n \quad (10)$$

$$\eta_i^+(y_i - (\beta_0 + x_i^T\beta) - \varepsilon_i) = 0, \quad for \quad i = 1,...,n \quad (11)$$

$$\eta_i^-(y_i - (\beta_0 + x_i^T\beta) + \varepsilon_i) = 0, \quad for \quad i = 1,...,n \quad (12)$$

$$\lambda(\sum_{i=1}^{n} sign(\beta_i)\beta_i - s) = 0 \quad (13)$$

$$\gamma_i\varepsilon_i = 0, \quad for \quad i = 1,...,n \quad (14)$$

For simplicity, we define:

- $R = \{i : y_i - (\beta_0 + x_i^T\beta) > 0\}$
- $L = \{i : y_i - (\beta_0 + x_i^T\beta) < 0\}$
- $E = \{i : y_i - (\beta_0 + x_i^T\beta) = 0\}$
- $V = \{j : \beta_j \neq 0\}$
- $\eta_i = \eta_i^+ - \eta_i^-$

From (10), we can get $-1 \le \eta_i \le 1$. If $i \in R$, from the first constraint in (6) we know $\varepsilon_i > 0$. Then, from (13) we can get $\gamma_i = 0$ and from (12) it can be found that $\eta_i^- = 0$. Therefore, from (10) $\eta_i^+ = 1$ is obtained. Similarly, rules for $i \in L$ can also be derived. When $i \in E$, ε_i must be 0; otherwise, $\eta_i^+ = 0$, $\eta_i^- = 0$ and $\gamma_i = 0$, which conflict with (10). These derived rules are listed as follows:

- $i \in R$: $\varepsilon_i > 0$, $\gamma_i = 0$, $\eta_i^+ = 1$, $\eta_i^- = 0$;
- $i \in L$: $\varepsilon_i > 0$, $\gamma_i = 0$, $\eta_i^+ = 0$, $\eta_i^- = 1$;
- $i \in E$: $\varepsilon_i = 0$, $\eta_i^+ + \eta_i^- \le 1$ (values unknown);

Theorem 1 *When* λ>0 and $|V| > 0$, t*he optimal solution of problem* (6) has $|E| = |V|$.

Proof: If $i \in L$ or *R*, then η_i is constant; only when $i \in E$, η_i is a variable. In (8) and (9), there are 1+|*E*| variables, η_i $(i \in E)$ and λ, and 1+|*V*| equations. Thus, we can get $|E| \ge |V|$; otherwise, generally speaking, (8) and (9) can not hold.

Since λ>0, from (13) we know $\sum_{i=1}^{n} sign(\beta_i)\beta_i - s = 0$. Also, we know that $y_i - (\beta_0 + x_i^T\beta) = 0$ for $i \in E$. For these equations, there are |*V*|+1 variables, β_0 and β_j $(j \in V)$, and |*E*|+1 equations. Similarly, we can get $|V| \ge |E|$.

Combining these together, we get |*E*|=|*V*|. □

When *s* increases an infinitesimal step, for continuity reasons, *E* and *V* can not change immediately, so we can get the following linear system:

$$\begin{cases} y_i - (\beta_0 + \sum_{j \in V} x_{ij}\beta_j) = 0, \quad i \in E \\ \sum_{j \in V} sign(\beta_j)\beta_j - s = 0 \end{cases} \quad (15)$$

It can be seen that β_0 and β_j $(j \in V)$ are linear in *s*. Therefore, when *E* and *V* are fixed, their derivatives *w.r.t. s* can be calculated as below:

$$\begin{cases} \frac{d\beta_0}{ds} + \sum_{j \in V} x_{ij}\frac{d\beta_j}{ds} = 0, \quad i \in E \\ \sum_{j \in V} sign(\beta_j)\frac{d\beta_j}{ds} = 1 \end{cases} \quad (16)$$

Since |*E*|=|*V*|, linear system (16) has a unique solution given that the corresponding matrix is nonsingular. With $\frac{d\beta_0}{ds}$ and $\frac{d\beta_j}{ds}$ $(j \in V)$, we can also solve the derivatives for residuals. Let $\xi_i = y_i - (\beta_0 + x_i^T\beta)$ represent the residual for point *i*, then its derivative *w.r.t. s* is:

$$\frac{d\xi_i}{ds} = -(\frac{d\beta_0}{ds} + \sum_{j \in V} x_{ij}\frac{d\beta_j}{ds}) \quad (17)$$

Since the residuals of points in *L* and *R* vary with *s*, one of them might reach 0 as *s* increases. In that case, we say an event happens. For each point $i \notin E$, we can determine when it will hit 0 by calculating:

$$\Delta s_i^1 = (y_i - (\beta_0 + \sum_{j\in V} x_{ij}\beta_j)) / (\frac{d\beta_0}{ds} + \sum_{j\in V} x_{ij}\frac{d\beta_j}{ds}) \quad (18)$$

Since *s* always increases, if $\Delta s_i^1 < 0$ for a point, we set it to $+\infty$, which means the point moves further away from 0 as *s* increases. The step size of *s* for this event can be calculated as:

$$\Delta s^1 = \min(\{\Delta s_i^1 \mid i \in L \;\; or \;\; R\}) \quad (19)$$

Another type of event can also happen, where a predictor variable in *V* has its coefficient reduced to zero. For each $j \in V$, we calculate:

$$\Delta s_j^2 = -\beta_j / \frac{d\beta_j}{ds} \quad (20)$$

Similarly, if $\Delta s_j^2 < 0$ for a predictor, then we need to set it to $+\infty$. The step size for this event is:

$$\Delta s^2 = \min(\{\Delta s_i^2 \mid j \in V\}) \quad (21)$$

Therefore, the overall step size is:

$$\Delta s = \min(\Delta s^1, \Delta s^2) \quad (22)$$

If $\Delta s^1 < \Delta s^2$, then we remove the corresponding point from *L* or *R* and add it into *E*; otherwise ($\Delta s^1 > \Delta s^2$), we remove the predictor variable from *V*. So, coefficients and residuals should be adjusted as follows:

- $\beta_0^{new} = \beta_0^{old} + \frac{d\beta_0}{ds}\Delta s$
- $\beta_j^{new} = \beta_j^{old} + \frac{d\beta_j}{ds}\Delta s, \quad j \in V$
- $\xi_i^{new} = \xi_i^{old} + \frac{d\xi_i}{ds}\Delta s, \quad i \in L \;\; or \;\; R$

Let E^{new} and V^{new} be the new set for *E* and *V*. No matter which event has happened, $|E^{new}| = |V^{new}| + 1$, which conflicts with *Theorem 1*. Thus, we need to either remove another point from *E* or add a new variable into *V* to make them balanced. It can be determined by solving the dual variables using (8) and (9).

Let L^{new} and R^{new} be the new set *L* and *R* after an event happens, and let $n_{L^{new}}$ and $n_{R^{new}}$ be the number of points in L^{new} and R^{new}, respectively. Now, (8) and (9) become:

$$\begin{cases} \sum_{i\in E^{new}} \eta_i + n_{R^{new}} - n_{L^{new}} = 0 \\ \sum_{i\in E^{new}} \eta_i x_{ij} + \sum_{i\in R^{new}} x_{ij} - \sum_{i\in L^{new}} x_{ij} = \lambda sign(\beta_j), \quad j \in V^{new} \end{cases} \quad (23)$$

In linear system (20), there are $|E^{new}|+1$ variables and $|V^{new}|+1$ equations. Since $|E^{new}| = |V^{new}|+1$, there is one degree of freedom. It can be seen that η_i ($i \in E^{new}$) is a linear function of λ, so we can get the derivative of η_i *w.r.t.* λ:

$$\begin{cases} \sum_{i\in E^{new}} \frac{d\eta_i}{d\lambda} = 0 \\ \sum_{i\in E^{new}} \frac{d\eta_i}{d\lambda} x_{ij} = sign(\beta_j), \quad j \in V^{new} \end{cases} \quad (24)$$

In linear system (24), there are unique solutions for $\frac{d\eta_i}{d\lambda}$'s ($i \in E^{new}$), given that the corresponding matrix is nonsingular. Note that the process of increasing *s* corresponds to reducing λ, so λ decreases throughout the algorithm. Therefore, when updating variable η_i's with $\eta_i^{new} = \eta_i^{old} + \frac{d\eta_i}{d\lambda}\Delta\lambda$, $\Delta\lambda$ always has negative value.

If η_i^{new} reaches -1, then point *i* should be removed from *E* and added into *L*; on the other hand, if η_i^{new} reaches 1, the point should be added into *R*. Therefore, for each $i \in E^{new}$, we can calculate:

- $\Delta\lambda_i^1 = (1 - \eta_i^{old}) / \frac{d\eta_i}{d\lambda}$
- $\Delta\lambda_i^2 = (-1 - \eta_i^{old}) / \frac{d\eta_i}{d\lambda}$

Note that only one of them can be negative. We define:

$$\Delta\lambda_i = \min(\Delta\lambda_i^1, \Delta\lambda_i^2) \quad (25)$$

From (9), it can be seen that for a zero-valued predictor *j* ($j \notin V$), if

$$\sum_{i\in E^{new}} \eta_i^{old} x_{ij} + \sum_{i\in R^{new}} x_{ij} - \sum_{i\in L^{new}} x_{ij} + \sum_{i\in E^{new}} \frac{d\eta_i}{d\lambda} x_{ij}\Delta\lambda = (\lambda + \Delta\lambda) sign(\beta_j)$$

then *j* should be included in *V* (β_j becomes nonzero). Therefore, for each $j \notin V$, we can calculate:

- $\Delta\tilde{\lambda}_j^1 = \dfrac{\sum_{i\in E^{new}} \eta_i^{old} x_{ij} + \sum_{i\in R^{new}} x_{ij} - \sum_{i\in L^{new}} x_{ij} - \lambda}{1 - \sum_{i\in E^{new}} \frac{d\eta_i}{d\lambda} x_{ij}}$
- $\Delta\tilde{\lambda}_j^2 = \dfrac{\sum_{i\in E^{new}} \eta_i^{old} x_{ij} + \sum_{i\in R^{new}} x_{ij} - \sum_{i\in L^{new}} x_{ij} + \lambda}{-1 - \sum_{i\in E^{new}} \frac{d\eta_i}{d\lambda} x_{ij}}$

And we define $\Delta\tilde{\lambda}_j$ as:

$$\Delta\tilde{\lambda}_j = \begin{cases} \Delta\tilde{\lambda}_j^1, & if \quad \Delta\tilde{\lambda}_j^2 > 0 \\ \Delta\tilde{\lambda}_j^2, & if \quad \Delta\tilde{\lambda}_j^1 > 0 \\ \max(\Delta\tilde{\lambda}_j^1, \Delta\tilde{\lambda}_j^2), & if \quad both \quad are \quad negative \end{cases} \tag{26}$$

The sign of β_j ($j \notin V$) can be determined by:

$$sign(\beta_j) = \begin{cases} +1, & if \quad \Delta\tilde{\lambda}_j = \Delta\tilde{\lambda}_j^1 \\ -1, & if \quad \Delta\tilde{\lambda}_j = \Delta\tilde{\lambda}_j^2 \end{cases} \tag{27}$$

Note that $\Delta\lambda_i$'s ($i \in E^{new}$) and $\Delta\tilde{\lambda}_j$'s ($j \notin V$) are all negative, so the step size $\Delta\lambda$ should be:

$$\Delta\lambda = \max(\{\Delta\lambda_i \mid i \in E^{new}\}, \{\Delta\tilde{\lambda}_j \mid j \notin V\}) \quad (28)$$

If $\Delta\lambda$ equals to one of $\Delta\lambda_i$'s ($i \in E^{new}$), then the corresponding point will be removed from *E*; if $\Delta\lambda$ equals to one of $\Delta\tilde{\lambda}_j$'s ($j \notin V$), the corresponding predictor should be added into *V*. After the adjustment, *E* and *V* will have the same number of elements, so we can calculate the step size for *s* using (16) to (19) again. The entire process is repeated until one of the following stopping criteria is met:

1. λ reduces to 0;
2. The maximal iteration number is reached;
3. All the predictor variables are included in V;

Finally, we describe how the algorithm gets started. The algorithm starts from *s*=0, which corresponds to $\lambda \to \infty$ in (5). It can be seen that $\hat{\beta}_j = 0$ *(j=1,2..,p)* and $V = \Phi$ at this time. Let $y_{(1)}, y_{(2)}, \ldots, y_{(n)}$ be the observations for the response variable in ascending order. If *n* is odd, let k^* be the index corresponding to $y_{(\lceil n/2 \rceil)}$. In this case, $\hat{\beta}_0 = y_{k^*}$ and $E = \{k^*\}$. If *n* is even, let k_1 and k_2 be the indices corresponding to $y_{(n/2)}$ and $y_{((n/2)+1)}$, respectively. Since either one of them can be added into *E*, we arbitrarily select k_1 and let $\hat{\beta}_0 = y_{k_1}$, $E = \{k_1\}$. *L* and *R* can be easily determined from residual values. For $i \in L$, $\eta_i = -1$ and for $i \in R$, $\eta_i = +1$. From (8) we can get: if *n* is odd, η_{k^*} =0; and if *n* is even, η_{k_1} = -1.

When $s \to 0^+$, one of the predictor variables will join *V*, but its magnitude still keeps 0. The index and sign of this variable are determined by:

- $l = \arg\max_j \{ | \sum_{i=1}^{n} \eta_j x_{ij} | \}$
- $sign(\beta_l) = sign(\sum_{i=1}^{n} \eta_j x_{il})$

At this point, λ reduces from ∞ to $| \sum_{i=1}^{n} \eta_j x_{il} |$, which can be seen from (9). Now, we have |*E*|=|*V*|=1 and the algorithm can proceed from here.

The complete algorithm is shown in Table 1:

Initialization:	Identify *L*, *R*, *E*; calculate $\hat{\beta}_0$ and η_i 's;
Step 1:	Solve Δs using (16) to (22); update *s*, $\hat{\beta}_0$, $\hat{\beta}_j$ $(j \in V)$ and ξ_i $(i \notin E)$; adjust *L*, *R*, *E* and *V* accordingly;
Step 2:	Solve $\Delta\lambda$ using (24) to (28); update λ, η_i ($i \in E^{new}$); adjust *L*, *R*, *E* and *V* accordingly;
Step 3:	If one of the stopping criteria is met, exit; otherwise goto Step 1.

Table 1. The RLAD algorithm

Figure 2 shows the solutions on the regularization path for a simple data set. The data consist of 20 samples and 10 predictor variables, among which 3 are true predictors. The horizontal axis represents the regularization parameter *s* and the vertical axis represents coefficient values. True predictors are plotted in solid lines and noise variables are plotted in dashed lines. The vertical dotted lines indicate where an event happens. From the figure, it can be seen that the continuous feature selection is achieved by changing *s*.

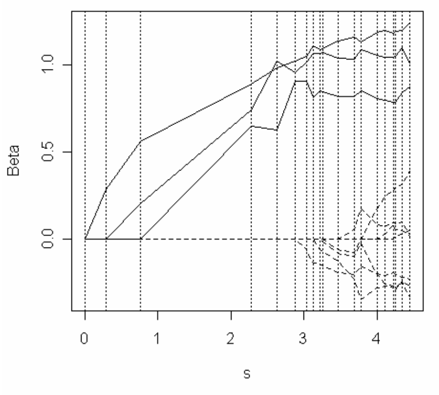


Figure 2. The solutions on the regularization path for a simple example

2.2. Computational cost

At step *k*, we need to solve two linear equations with size $|V^k|+1$ (or $|E^k|+1$), where V^k and E^k represent set *V* and *E* at step *k*. So, the computational complexity at each step seems to be $O(|V^k|^3)$. However, if we compare the linear equation systems at two consecutive steps, two matrices are differed by only one row or column. Therefore by inverse updating and downdating, the computational complexity at each step can be reduced to $O(|V^k|^2)$ (or $O(|E^k|^2)$), which is upper bounded by $O(\min(p,n)^2)$. It is difficult to predict how many steps are on the entire regularization path for an arbitrary problem, but our experience suggests that $O(\log(np))$ is a reasonable estimate. Therefore, the overall computational complexity for the RLAD algorithm is $O(\log(np)\min(p,n)^2)$.

Table 2 and 3 show the average computational time and number of steps for solving the entire regularization path with various *n* (sample size) and *p* (number of predictors). We implement the algorithm in *R*, but since the inverse updating and downdating part has not been implemented, the computational complexity of our code is $O(\log(np)\min(p,n)^3)$ right now. The experiments were conducted on a laptop with 1.66 GHz CPU and 1 GB memory. We repeat each setting 20 times and every time *X* and *y* are randomly generated. The figures in the following tables represent the average computational time based on 20 replicates and figures in parenthesis indicate the average number of steps.

As an interpreted language, *R* is relatively slow in executing programs compared with compiled languages and our code has not implemented inverse updating/downdating; however, from the following tables it can be seen that the computational time is basically acceptable for moderately large problems. The computational time can be significantly reduced after it is coded in compiled languages with inverse updating/downdating being implemented. According to our experience, solving a problem with the same size using simplex method often takes more time; the interior-point method is much faster, which takes around 1/10 of the time used by the RLAD algorithm. However these methods can only give the solution for a single value of *s*, while our algorithm solves the entire regularization path, which often consists of hundreds of solutions. For regularization method, the value of regularization parameter is always crucial to the prediction performance of the model, so the RLAD algorithm is advantageous given the solutions it provides and the time it uses.

	n		
	n=5,000	**n=10,000**	**n=20,000**
p=10	1.8 sec (200.2)	3.71sec (238.0)	11.5 sec (421.8)
p=20	4.6 sec (380.4)	12.5 sec (550.7)	32.1 sec (813.2)
p=40	15.8 sec (752.8)	41.6 sec (1166.2)	89.8 sec (1373.6)

Table 2. Computational time for *n* > *p*

	n		
	n=50	**n=100**	**n=200**
p=1000	4.3 sec (246.9)	16.6 sec (685.5)	90.9 sec (1896.2)
p=2000	8.3 sec (257.4)	34.4 sec (791.2)	154.6 sec (2048.2)
p=4000	20.4 sec (310.2)	71.8 sec (848.6)	302.2 sec (2252.6)

Table 3. Computational time for *n* < *p*

3. Simulations

In this section, we use simulations to demonstrate the prediction performance of our algorithm as well as its feature selection function. For simplicity, we ignore the intercept β_0 and assume the predictor variables are standardized. We assume the predictor vector x has multivariate normal distribution $N(\vec{0},\Sigma)$ and the response variable is calculated by $y = x^T\beta + \varepsilon$, where ε is noise. First, we generate *n* data points as training data and run the RLAD algorithm. Then 20,000 testing data are generated using the same distribution and prediction errors of the obtained models can be calculated, which are measured by mean absolute deviations. Since the size for the testing data is large enough, we can treat it as the entire population and the associated testing error as the true prediction error. Also, since we know the true model β, we can calculate the optimal prediction error for comparison purpose. Figure 3 shows the models (coefficient values) on the regularization path and prediction errors associated with these models. The solid curve represents the prediction errors on the testing data. The solid vertical line indicates where the minimal prediction error is achieved. The dashed horizontal line represents the error obtained from the true model. And the dotted curves are coefficients associated with different values of *s*. Therefore, it can be seen that the best model generated by the RLAD algorithm has its testing error pretty close to the optimal error.

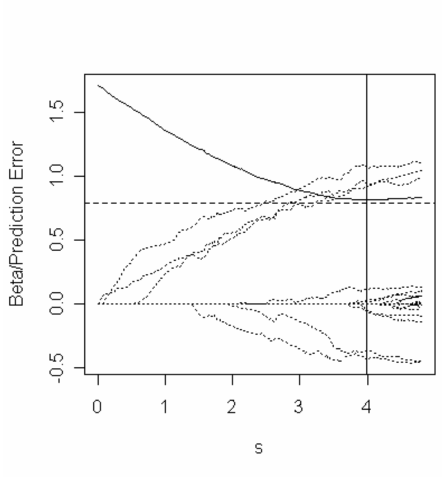


Figure 3. Models on the regularization path and their prediction errors

In simulations, we essentially use the entire population for model selection. In practice, this can not be the case. For real problems, we usually divide the data into two parts, run the algorithm on one part and test the models on the other; we can also use the cross-validation or GACV [12] approach. However, these methods mix the model generation and model selection together, which add uncertainty to the final results. In other words, these methods would not allow us to tell whether the poor prediction performance comes from bad models or the model selection process. Therefore, our experiment setting can precisely measure the performance of the models without interference of the model selection process.

In the experiments, we let the predictor vector x follow the multivariate normal distribution $N(\vec{0},\Sigma)$, where

$$\Sigma = \begin{bmatrix} \overbrace{1 \ \ \rho \ \ \ldots \ \ \rho}^{q} & 0 & \ldots & 0 \\ \rho \ \ 1 \ \ \ldots \ \ \rho & 0 & \ldots & 0 \\ \ldots & & & \\ \rho \ \ \ldots \ \ \ 1 & 0 & \ldots & 0 \\ & \ldots & & \\ 0 \ \ 0 \ \ \ldots & 0 & \ldots & 1 \end{bmatrix}$$

Thus, the first q variables are correlated and the rest are independent to each other. The response variable is calculated as $y = x^T\beta + \varepsilon$, where

$$\beta = [\beta_1, \beta_2, \ldots, \beta_L, 0, \ldots 0]^T$$

The first L variables are true predictors and ε is noise. For ε, we use both the normal distribution $N(0,\sigma)$ and the double exponential distribution with parameter λ in our experiments. The following tables show the experiment results for various settings. For each setting, the experiment is repeated 20 times. The figures in the tables are mean values and figures in the parenthesis are standard deviations. "Err^{Opt}" and "Err^{RLAD}" represent the prediction error for the true model and the best model obtained from the RLAD algorithm, respectively. "$\text{N}^{\text{Pr}ed}$" and "N^{True}" measure the automatic feature selection function. "$\text{N}^{\text{Pr}ed}$" represents the number of nonzero predictors that are included in the model and "N^{True}" is the number of true predictors which are identified by the RLAD model.

The experiments test the eight combinations of the following settings: $n>p$ vs. $n<<p$, normal noise vs. double exponential noise and independent predictors vs. highly correlated predictors. From the results, we can see that the performance of the RLAD algorithm is not affected much by the noise distribution. Its performance is not affected by the existence of highly correlated predictors, either. Multicollinearity is a big problem for linear regression. When predictors are highly correlated, linear regression tends to generate erroneous p-values, so true predictors can not be correctly identified. However, nearly all the true predictors are discovered by the RLAD algorithm when multicollinearity problem exists. The performance is impaired when $n<<p$ and the signal-to-noise ratio is extremely low. This situation is well known to be difficult and there has not been a good solution for solving it. Linear regression can not work when there are more predictors than the sample size. However, on average the RLAD algorithm can capture 60~90% of the true predictors and its prediction error is acceptable even when p is significantly larger than n.

Err^{Opt} = 0.80	$\rho=0,\ \ \sigma=1,\ \ \beta=(1,1,1,0.5,0.5,0,\ldots 0)$		
	Err^{RLAD}	**$\text{N}^{\text{Pr}ed}$**	**N^{True}**
n=200 p=10	0.82 (0.02)	8.8 (1.1)	5.0 (0)
n=100 p=10	0.84 (0.03)	7.8 (1.4)	5.0 (0)
n=50 p=10	0.89 (0.04)	7.5 (0.9)	5.0 (0.2)

Table 4. Results for normal noise, independent predictors and n>p

Err^{Opt} = **0.80**	$\rho = 0, \quad \sigma = 1, \quad \beta = (1,1,1,0.5,0.5,0,....0)$		
	Err^{RLAD}	**N**^{Pred}	**N**^{True}
n=100 p=100	0.92 (0.05)	23.9 (8.5)	5.0 (0.0)
n=50 p=100	1.09 (0.13)	25.3 (12.7)	4.8 (0.6)
n=25 p=100	1.25 (0.19)	16.1 (4.01)	3.8 (1.01)

Table 5. Results for normal noise, independent predictors and n<p

Err^{Opt} = **0.79**	$\rho = 0.8, \quad q = 10, \quad \sigma = 1, \quad \beta = (1,1,1,1,1,0,....0)$		
	Err^{RLAD}	**N**^{Pred}	**N**^{True}
n=200 p=10	0.82 (0.01)	7.8 (1.1)	5.0 (0)
n=100 p=10	0.84 (0.03)	7.4 (1.5)	5.0 (0)
n=50 p=10	0.86 (0.03)	7.4 (1.6)	5.0 (0.2)

Table 6. Results for normal noise, correlated predictors and n>p

Err^{Opt} = **0.79**	$\rho = 0.8, \quad q = 10, \quad \sigma = 1, \quad \beta = (1,1,1,1,1,0,....0)$		
	Err^{RLAD}	**N**^{Pred}	**N**^{True}
n=100 p=100	0.87 (0.04)	12.7 (4.5)	5.0 (0)
n=50 p=100	0.97 (0.08)	15.2 (5.6)	4.8 (0.4)
n=25 p=100	1.07 (0.11)	14.9 (4.6)	4.6 (0.6)

Table 7. Results for normal noise, correlated predictors and n<p

Err^{Opt} = **1.00**	$\rho = 0, \quad \lambda = 1, \quad \beta = (1,1,1,0.5,0.5,0,....0)$		
	Err^{RLAD}	**N**^{Pred}	**N**^{True}
n=200 p=10	1.01 (0.01)	8.3 (1.3)	5.0 (0)
n=100 p=10	1.04 (0.04)	8.1 (1.0)	5.0 (0)
n=50 p=10	1.13 (0.08)	8.4 (1.2)	5.0 (0)

Table 8. Results for double exponential noise, independent predictors and n>p

Err^{Opt} =	$\rho = 0, \quad \lambda = 1, \quad \beta = (1,1,1,0.5,0.5,0,....0)$		
	Err^{RLAD}	**N**^{Pred}	**N**^{True}
n=100 p=100	1.14 (0.07)	20.8 (6.07)	4.9 (0.3)
n=50 p=100	1.32 (0.13)	16.9 (5.7)	4.3 (0.7)
n=25 p=100	1.61 (0.12)	12.7 (7.1)	3.0 (1.1)

Table 9. Results for double exponential noise, independent predictors and n<p

Err^{Opt} = **1.01**	$\rho = 0.8, \quad q = 10, \quad \lambda = 1, \quad \beta = (1,1,1,1,1,0,....0)$		
	Err^{RLAD}	**N**^{Pred}	**N**^{True}
n=200 p=10	1.03 (0.02)	13.8 (4.4)	5.0 (0)
n=100 p=10	1.07 (0.05)	13.7 (5.7)	5.0 (0.2)
n=50 p=10	1.19 (0.10)	14.1 (6.6)	4.9 (0.4)

Table 10. Results for double exponential noise, correlated predictors and n>p

Err^{Opt} = **1.01**	$\rho = 0.8, \quad q = 10, \quad \lambda = 1, \quad \beta = (1,1,1,1,1,0,....0)$		
	Err^{RLAD}	**N**^{Pred}	**N**^{True}
n=100 p=100	1.09 (0.05)	16.4 (6.9)	5.0 (0)
n=50 p=100	1.19 (0.11)	12.9 (4.7)	4.7 (0.5)
n=25 p=100	1.37 (0.28)	12.3 (5.4)	4.3 (0.8)

Table 11. Results for double exponential noise, correlated predictors and n<p

4. Image Reconstruction

Images can often be contaminated when they are captured or transmitted. In this section, we apply the RLAD algorithm to solve the image reconstruction problem. The problem has been well studied and there are several categories of methods. When the distribution of the image is unknown, smoothing techniques [1,4] are often used. However, when people have prior knowledge about this distribution, principal component analysis (PCA) method [6, 8, 11] is often preferred. In PCA method, the contaminated image is projected onto the space of principal components, which are calculated from the image database. Our experiment data come from the MPI face database, which contains 200 face images.

We randomly pick an image from the MPI database and contaminate 20% of its pixels. For each selected pixel, we assume its information is completely damaged, so its color is set to white. With this setting, the noise distribution is not normal, but heavy-tailed. As shown later, since the RLAD algorithm uses the robust loss function, its performance is not affected. Figure 4 shows the original image and the corrupted one. It can be seen that with 20% contaminated pixels the original face can hardly be recognized by visual inspection.

Figure 4. The true and the corrupted images

From the remaining 199 images, we calculate their principal components (PC). Figure 5 shows the images for some of them. The PCs are orthogonal to each other and each of them can capture some characteristics for a face. The original image will be reconstructed by summing the weighted PCs together. Since the first 20 eigenvalues are significantly larger than the rest, their corresponding PCs are used in our experiment.

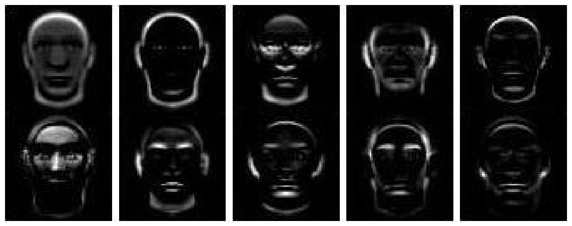


Figure 5. The 10 largest principal components

To set up the baseline, we use both the least square projection and the *idealistic* method. In the least square projection, the corrupted image is projected to the PC space in the sense that the residual sum of squares is minimized. For the *idealistic* method, since we know the exact locations for noise pixels, we can remove them and project only the good pixels onto the corresponding parts of the PC space. Then the obtained weights are used for reconstructing the whole image. This represents the idealistic situation, because in reality the locations for noise can not be easily determined. Therefore, the image generated by the *idealistic* method is often used as the golden standard for image reconstruction. From our literature reviews, this standard can not be achieved by existing methods. In [11], the proposed method can arrive at the midpoint between the least square projection and the *idealistic* method. Figure 6 shows the results obtained by the least square projection and the idealistic approach, respectively.

Figure 6. The reconstructed images by the least square projection and the *idealistic* method

Assume that an image consists of n pixels. Let y represent the corrupted image and X be an n by p matrix, whose columns correspond to the p principal components. The principal components are standardized in our experiments. Inspired by [11], which uses the L_∞ norm as the loss function and L_1 as the penalty, we propose the following model for image reconstruction:

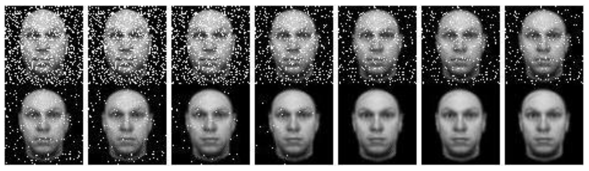
$$\begin{aligned} &\min_{\beta_0,\beta_1,\beta_2} \| y-(\beta_0+\overline{X}\begin{bmatrix}\alpha\\ \beta\end{bmatrix})\|_1 \\ &s.t. \quad \|\begin{bmatrix}\alpha\\ \beta\end{bmatrix}\|_1 \le s \end{aligned} \qquad (29)$$

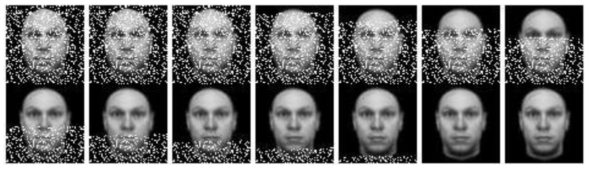
where $\overline{X}=[I_{nxn} \quad X_{nxp}]$ and α and β have length n and p, respectively. Therefore, the RLAD algorithm can solve its solutions for every value of s.

α is used to pinpoint the noise pixels: the nonzero components of α correspond to the noise pixels. β is used to identify the relevant PCs, where the zero-valued components of β eliminate the irrelevant PCs. With this setting, for each s we can obtain two reconstructed images: $y-\alpha$ and $\beta_0+X\beta$. Image $y-\alpha$ is generated in a pixel-by-pixel fashion, where noise pixels are automatically identified and only these pixels are modified. Image $\beta_0+X\beta$ is obtained completely from the PCs. The RLAD algorithm is used to solve problem (29). However, we did not solve the entire regularization path. Once noise pixels are not visually observable from the output images, we stop the program. Figure 7 shows the image series on the regularization path. The horizontal axis represents the number of steps used, which has been standardized.

Note that two or more predictor variables might tie with each other when the algorithm is checking which one should be added into the active set V for the next step. This is called the degeneracy problem in linear programming. When it occurs, we have two choices: randomly picking a variable or picking one with the smallest index. Figure 7 (a) and (b) show the $y-\alpha$ images corresponding to these two choices. It is interesting to note that they generate completely different denoising behaviors. In (a), the noise density is gradually reduced until all of the noise pixels are eliminated, but before that we can not find a zone which is completely noise free. However, in (b) an interesting phenomenon can be observed, where noise pixels are removed from top to bottom, since pixels on top have smaller indices. Figure 7 (c) shows the images for $\beta_0 + X\beta$.

From figure 7, it can be seen that $\beta_0 + X\beta$ images are smooth and have high quality from the beginning of the algorithm, but they do not change much as the algorithm proceeds. On the other hand, in the pixel-by-pixel reconstructed images, noise pixels are gradually identified and removed. At the end, both image series can achieve pretty good quality, which look exactly the same as the image generated by the idealistic method.


(a) The pixel-by-pixel reconstructed image series (random)


(b) The pixel-by-pixel reconstructed image series (sequential)

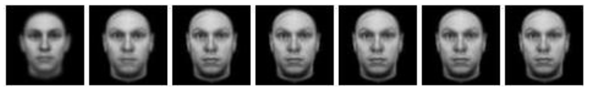
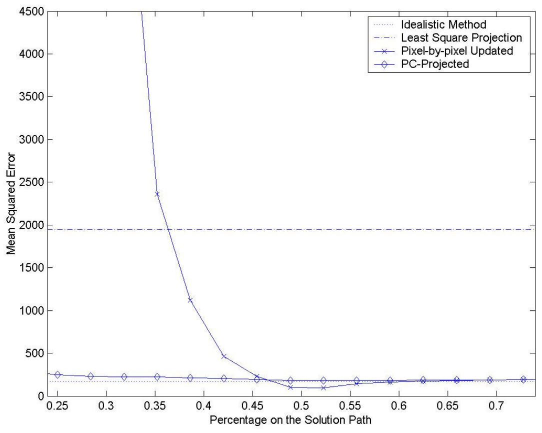

(c) The PC projection image series

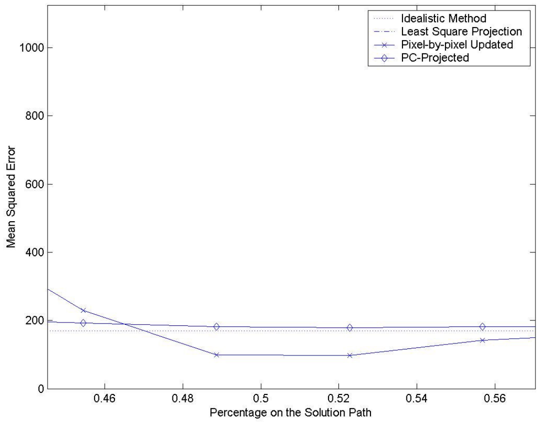
Figure 7. Output images from the RLAD algorithm

Figure 8 shows the mean squared errors (MSE) between the reconstructed images and the true image. The two solid curves represent the MSE for image $y-\alpha$ and $\beta_0 + X\beta$, respectively. In figure 8 (a), the upper horizontal line (dashed) indicates the performance for the least square projection and the lower horizontal line (dotted) indicates that for the *idealistic* method. The $\beta_0 + X\beta$ series finally approach the golden standard, while the $y-\alpha$ images can even exceed it. This is a surprisingly good result, because the golden standard can rarely be achieved by previous methods.

From figure 8, we can see the importance of the parameter tuning algorithm. The performance of $y-\alpha$ images is sensitive to the value of the regularization parameter. Therefore, the best setting can be easily missed if we use the trial-and-error approach for parameter tuning.



(a) Performance Curve (big picture)



(b) Performance Curve (zoomed picture)

Figure 8. Image Denoising Performance Curve

4. Conclusions and Future Work

In this paper, we combine the LAD regression and the LASSO method together and propose the RLAD regression model. Its benefits are three-fold: 1) by using the robust loss function, its results are not sensitive to outliers; 2) by using the L_1 norm penalty, both the variance of the RLAD estimator and the prediction error are reduced; 3) it can perform automatic feature selection, where coefficients for irrelevant features are shrunk to zero. To facilitate parameter tuning, we develop an efficient algorithm, which can solve all the solutions on the regularization path. Simulations are designed to demonstrate the performance of the RLAD algorithm. The algorithm is also tested in a real-world application, the image reconstruction problem, and it achieves surprisingly good performance.

One limitation of the RLAD model is that model selection can not be easily performed. Since there are no assumptions for the noise distribution, asymptotic theories are difficult to develop. Recall in our simulations, model selection is performed on the entire population and in the image reconstruction problem it is achieved by visual inspection. Although we can perform model selection using the cross-validation method, it is a computationally expensive method. Therefore, designing a good criterion for selecting model is one of the directions for our future work. Another direction is to adopt other robust functions, such as the Huber function, as the loss function.

References

[1] M. Black and A. Rangarajan, “On the unification of line processes, outlier rejection, and robust statistics with applications in early vision,” International Journal of Computer Vision, 25 (19):57-92, 1996

[2] A. Giloni and M. Padberg, "Alternative methods of linear regression", Mathematical and Computer Modelling 35, 361-374, 2002

[3] A. Giloni and M. Padberg, "The finite sample breakdown point of L1-regression," SIAM Journal on Optimization, 14, 1028-1042, 2004

[4] A. Hamza and H. Krim, “Image denoising: A nonlinear robust statistical approach,” IEEE Transaction on Signal Processing, 49 (12): 3045-3054, 2001

[5] X. He, J. Jurechova, R. Koenker, and S. Portnoy, "Tail behavior of regression estimators and their breakdown points," Econometrica, 58, 1195-1214, 1990

[6] S. Mika, B. Scholkopf, A. Smola, K. Muller, M. Scholz and G. Ratsch, “Kernel PCA and de-noising in feature spaces,” Advances in Neural Information Processing Systems, vol 11, 537-542, MIT Press, 1999

[7] I. Mizera and C.H. Muller, "The influence of the design on the breakdown points of L1-type M-esimators," Advances in Model-Oriented Design and Analysis, Physica-Verlag, Heidelberg, 193-200, 2001

[8] T. Takahashi and T. Kurita, “Robust de-noising by kernel PCA,” Artificial Neural Networks, 727-732, Springer Verlag, 2002

[9] W.F. Sharpe, “Mean-absolute-deviation characteristic lines for securities and portfolios,” Management Science 18, B1-B13, 1971

[10] R. Tibshirani, “Regression shrinkage and selection via the lasso,” Journal of Royal Statistical Society, Vol. 58, No.1, 267-288, 1996

[11] K. Tsuda and G. Ratsch, “Image reconstruction by linear programming,” Nerual Information Processing Systems (NIPS), 2003

[12] M. Yuan, “GACV for quantile smoothing splines,” Computational Statistics and Data Analysis, 2005