Journal of Applied Statistics

ISSN: 0266-4763 (Print) 1360-0532 (Online) Journal homepage: http://www.tandfonline.com/loi/cjas20

Robust sparse regression by modeling noise as a mixture of gaussians

Shuang Xu & Chun-Xia Zhang

To cite this article: Shuang Xu & Chun-Xia Zhang (2019): Robust sparse regression by modeling noise as a mixture of gaussians, Journal of Applied Statistics, DOI: 10.1080/02664763.2019.1566448

To link to this article: https://doi.org/10.1080/02664763.2019.1566448

Published online: 11 Jan 2019.

JOURNAL OF APPLIED STATISTICS
https://doi.org/10.1080/02664763.2019.1566448

Taylor & Francis
Taylor & Francis Group

Robust sparse regression by modeling noise as a mixture of gaussians

Shuang Xu and Chun-Xia Zhang

School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, People's Republic of China

ABSTRACT

Regression analysis has been proven to be a quite effective tool in a large variety of fields. In many regression models, it is often assumed that noise is with a specific distribution. Although the theoretical analysis can be greatly facilitated, the model-fitting performance may be poor since the supposed noise distribution may deviate from real noise to a large extent. Meanwhile, the model is also expected to be robust in consideration of the complexity of real-world data. Without any assumption about noise, we propose in this paper a novel sparse regression method called MoG-Lasso to directly model noise in linear regression models via a mixture of Gaussian distributions (MoG). Meanwhile, the L_1 penalty is included as a part of the loss function of MoG-Lasso to enhance its ability to identify a sparse model. As for the parameters in MoG-Lasso, we present an efficient algorithm to estimate them via the EM (expectation maximization) and ADMM (alternating direction method of multipliers) algorithms. With some simulated and real data contaminated by complex noise, the experiments show that the novel model MoG-Lasso performs better than several other popular methods in both '$p > n$' and '$p < n$' situations, including Lasso, LAD-Lasso and Huber-Lasso.

ARTICLE HISTORY
Received 27 March 2018
Accepted 3 January 2019

KEYWORDS
Robust regression; penalized regression; variable selection; mixture of Gaussians; lasso

1. Introduction

In statistics and many other disciplines, linear regression [24] is always a hot topic because of its simplicity and good performance. Given some observed data $(\boldsymbol{y}, \boldsymbol{X}) = \{(y_i, \boldsymbol{x}_i^{\mathrm{T}})\}_{i=1}^n = \{(y_i, x_{i1}, \ldots, x_{ip})\}_{i=1}^n$ of the response variable Y and the covariates $X_1, \ldots, X_p$, it is usually assumed that $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ in which $\boldsymbol{\beta}$ is a p-dimensional unknown coefficient vector. As for the noise term $\boldsymbol{\varepsilon}$, researchers often hypothesize that it comes from a Gaussian distribution [24]. In this situation, it is well-known that the ordinary least-squares (OLS) method can provide the optimal estimation of $\boldsymbol{\beta}$, i.e.

$$\boldsymbol{\beta}^{\mathrm{ols}} = \arg\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2. \tag{1}$$

When facing with high-dimensional data, especially those with $p \geq n$, Tibshirani [28] developed Lasso (least absolute shrinkage and select operator) to simultaneously achieve

CONTACT Chun-Xia Zhang cxzhang@mail.xjtu.edu.cn School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, People's Republic of China

a sparse solution and high estimation accuracy by minimizing the L_2-norm loss plus the L_1-norm penalty on $\boldsymbol{\beta}$, namely,

$$\boldsymbol{\beta}^{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1, \tag{2}$$

where λ is the penalty parameter which is generally tuned by cross validation. Up to now, Lasso has been studied extensively in both theory and applications [14,30,34].

It is well known that both OLS and Lasso employ the L_2-norm loss function, which, from the Bayesian viewpoint, coincides with the maximum likelihood estimation when noise is sampled from a Gaussian distribution. Unfortunately, the noise of real-world data usually does not come from a Gaussian distribution, or is corrupted by outliers. Therefore, OLS and Lasso may fail to process real-world data [4]. Given a high-dimensional dataset corrupted by outliers, the Lasso estimator may deviate from the true $\boldsymbol{\beta}$ and lead to great residuals. In this case, cross validation is very likely to select a null model whose residual sum of squares (RSS) is $\text{RSS}_0 = \sum_{i=1}^{n}(y_i - \bar{y})^2$ since other models' RSS is larger than RSS_0. Therefore, *robust sparse regression* has attracted more and more attention of researchers from the communities of statistics and machine learning [6,10,23,31–33,36].

Candes and Tao [5] proposed the Dantzig selector, that is

$$\boldsymbol{\beta}^{\text{Dantzig}} = \arg\min_{\boldsymbol{\beta}} ||\boldsymbol{X}^{\text{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})||_\infty + \lambda||\boldsymbol{\beta}||_1, \tag{3}$$

where $||\cdot||_\infty$ denotes the L_∞ norm, the maximum absolute value of a vector. If covariates and residual have been normalized, $\boldsymbol{X}^{\text{T}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$ is actually the correlation coefficient between covariates and the residual. Compared with L_2-norm loss function, correlation coefficient, to a certain extent, is more robust, while it still cannot cope with the cases of *heavy-tailed noise* well. To improve the robustness of a regression model, one way is to directly hypothesize that noise is from a heavy-tailed distribution [33]. For example, the loss function becomes the sum of absolute deviations when noise is distributed as a Laplace distribution. Hence, we obtain the LAD (least absolute deviation) estimator as

$$\boldsymbol{\beta}^{\text{lad}} = \arg\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_1 = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} |r_i|, \tag{4}$$

where r_i is defined as $r_i = y_i - \boldsymbol{x}_i^{\text{T}}\boldsymbol{\beta}$. Because the objective function in (4) is non-smooth, the traditional methods, including Taylor expansion argument, cannot be directly applied to study the asymptotic properties of LAD estimator. Thereafter, researchers have devoted to establishing the $\sqrt{n}$-consistency and asymptotic normality of LAD estimator [2,20]. Wang *et al.* [33] developed LAD-Lasso, proved its $\sqrt{n}$-consistency and the oracle property. They also conducted some experiments to show its robustness. However, their experiments only focused on small datasets of $p = 8$, $n = 50{,}100{,}200$ with mild heavy-tailed noise, including Laplace, standard t-distribution with degrees of freedom 3 and 5. Although it is interesting to study the behavior of LAD-Lasso in '$p > n$' situations with severe heavy-tailed noise, to the best of our knowledge, however, little work is done to address this issue. At the same time, the optimization of LAD-Lasso is more difficult than Lasso, owing to the non-smoothness of L_1-norm loss. Furthermore, the performance of LAD-Lasso may be poor if the supposed Laplace distribution is far from the true noise distribution.

Another popular way is to replace the L_2-norm loss function by the loss functions used in robust statistics. The Huber loss [16,17] was an important tool in classical robust statistics and the Huber regression is

$$\boldsymbol{\beta}^{\text{huber}} = \arg\min_{\boldsymbol{\beta}} f_{\text{huber}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}), \tag{5}$$

where

$$f_{\text{huber}}(\boldsymbol{t}) = \sum_{i=1}^{n} \frac{t_i^2}{2\gamma}\mathbb{I}(|t_i| \le \gamma) + \left(|t_i| - \frac{\gamma}{2}\right)\mathbb{I}(|t_i| > \gamma). \tag{6}$$

Here, $\mathbb{I}(\cdot)$ denotes the indicator function and γ is a hyper-parameter. Note that $f(\cdot)$ in (6) is a piecewise function, which imposes L_1 norm for large residuals in order to weaken the influence of outliers. Evidently, the Huber loss is more flexible than L_1-norm loss and it will converge to L_1 norm if γ goes to 0. Rosset and Zhu [25] considered the problem minimizing the Huber loss plus a Lasso-type penalty, namely, Huber-Lasso. Very recently, Yi and Huang [36] developed a fast algortithm (semismooth Newton coordinate descent) to handle sparse Huber regression. Nevertheless, if the hyper-parameter γ is not pre-defined well, Huber–Lasso is very likely to lose robustness. Another promising tool in robust statistics is the trimmed squares loss function introduced by Rousseeuw [26], the main idea of which is to delete large squared residuals. In particular, the least trimmed squares estimator (LTS) is defined as

$$\boldsymbol{\beta}^{rmlts} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{h} r_i^o, \tag{7}$$

where r_i^o is the ordered squared residuals satisfying $r_1^o \le \cdots \le r_n^o$ and h denotes the initial guess of the number of uncontaminated samples. Alfons and Gelper [1] came up with the LTS-Lasso method in which the L_1-norm penalty is imposed on $\boldsymbol{\beta}$. They proved that LTS-Lasso is with a high breakdown point and showed that LTS-Lasso is robust to outliers. Besides the above-mentioned methods, Khan *et al.* [18] developed a robust version of least angle regression algorithm via letting a robust type of correlation coefficient be the estimate of $\boldsymbol{\beta}$, instead of the Pearson correlation coefficient.

From the Bayesian viewpoint, the ideal case is to directly model noise without any assumption. At present, however, it is still an open issue about how to design a robust regression model with an unknown noise distribution. To solve this issue, by modeling noise with a mixture of Gaussian distributions (MoG), we develop a robust sparse regression called MoG-Lasso. Meanwhile, an L_1 penalty is included as a part of the loss function of MoG-Lasso so that a sparse model can be identified. Typically, we usually employ two Gaussian components, in which the one with low variance models inliers, while the other one with high variance models outliers. On the one hand, the parameters involved in MoG can be easily estimated via expectation maximization (EM) algorithm according to the maximum likelihood principle. In the M-step of the EM algorithm, on the other hand, we solve a weighted Lasso problem in which the weight of each observation is learned automatically. In this manner, the influences of outliers is weakened by assigning small weight to outliers. Experiments show that in situations with complex noise, MoG-Lasso outperforms several counterparts (including Lasso [28], LAD-Lasso [33], Huber–Lasso [36]) in terms of both variable selection accuracy and prediction accuracy.

The rest of the paper is organized as follows. The MoG-Lasso model is proposed in Section 2. In Section 3, we examine and compare the performance of MoG-Lasso with some existing methods by conducting experiments on both simulated and real-world datasets. Finally, discussions and some conclusions are presented in Sections 4 and 5, respectively.

2. Sparse linear regression with MoG noise

2.1. Penalized regression with MoG noise

It has been shown that MoG is able to approximate any continuous distribution well [21]. When the noise distribution is unknown, it is natural to directly model noise by an MoG [11,22] as

$$p(\varepsilon_i) \overset{\text{i.i.d.}}{\sim} \text{MoG}(\varepsilon_i|K, \boldsymbol{\Pi}, \mathbf{0}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \pi_k p(\varepsilon_i|0, \sigma_k^2), \tag{8}$$

where K stands for the number of Gaussian components, $\boldsymbol{\Sigma} = \{\sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2\}$, $\boldsymbol{\Pi} = \{\pi_1, \pi_2, \ldots, \pi_K\}$ and $p(\varepsilon_i; 0, \sigma_k^2)$ represents the probability density function of Gaussian distribution with mean 0 and variance σ_k^2. In (8), $\pi_k \geq 0$ is the mixing proportion and there is $\sum_{k=1}^{K} \pi_k = 1$. With the above notations and assumptions, the log-likelihood function is

$$\ell(\boldsymbol{\Omega}) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \pi_k p(y_i|\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}, \sigma_k^2) \right], \tag{9}$$

where $\boldsymbol{\Omega} = \{\boldsymbol{\beta}, \boldsymbol{\Pi}, \boldsymbol{\Sigma}\}$ include the unknown parameters. In order to obtain a sparse solution of $\boldsymbol{\beta}$, we adopt a penalized log-likelihood function $\tilde{\ell}(\boldsymbol{\Omega}) = \ell(\boldsymbol{\Omega}) + \lambda g(\boldsymbol{\beta})$, where λ is a tuning parameter and $g(\boldsymbol{\beta})$ is the penalty function. For ease of computation, we introduce a latent vector $\boldsymbol{l}_i = (l_{i1}, l_{i2}, \ldots, l_{iK})^{\mathrm{T}}$ for each observation where $l_{ik} \in \{0, 1\}$ and $\sum_{k=1}^{K} l_{ik} = 1$. Evidently, $\boldsymbol{l}_i$ follows a multinomial distribution, i.e. $\boldsymbol{l}_i \sim \mathcal{M}(\pi_1, \pi_2, \ldots, \pi_K)$. As a result, the penalized log-likelihood function can be rewritten as

$$\tilde{\ell}(\boldsymbol{\Omega}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \log[\pi_k p(y_i|\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}, \sigma_k^2)]^{l_{ik}} + \lambda g(\boldsymbol{\beta}). \tag{10}$$

Owing to the existence of the latent variables l_{ik} in $\tilde{\ell}(\boldsymbol{\Omega})$, the EM algorithm [9] can be employed to infer our model because of its efficiency to deal with incomplete data. The detailed EM algorithm can be summarized as follows.

E-step: Compute the conditional expectation of l_{ik} as

$$E(l_{ik}|y_i) = \frac{\pi_k p\left(y_i|\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}, \sigma_k^2\right)}{\sum_{k=1}^{K} \pi_k p\left(y_i|\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}, \sigma_k^2\right)} \equiv \gamma_{ik}, \quad i = 1, \ldots, n; k = 1, \ldots, K. \tag{11}$$

In what follows, we let $\boldsymbol{\Gamma} = (\gamma_{ik})_{n\times K}$ and $N_k \equiv \sum_{i=1}^{n} \gamma_{ik}$. After some derivations (see details in Appendix), we can get the so-called Q-function as

$$Q(\boldsymbol{\Omega}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \left[\log \pi_k - \log \sigma_k^2 - \frac{(y_i - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})^2}{2\sigma_k^2} \right] + \lambda g(\boldsymbol{\beta}). \tag{12}$$

M-step: Maximize $Q(\boldsymbol{\Omega})$ by alternatively updating $\boldsymbol{\Pi}, \boldsymbol{\Sigma}$ and $\boldsymbol{\beta}$. First, the updates for the MoG parameters π_k and σ_k^2 (the detailed derivation is provided in Appendix) can be obtained as

$$\pi_k = \frac{N_k}{n}, \quad \sigma_k^2 = \frac{1}{N_k}\sum_{i=1}^{n}\gamma_{ik}\left(y_i - \boldsymbol{x_i}^{\mathrm{T}}\boldsymbol{\beta}\right)^2, \quad k = 1, 2, \ldots, K. \tag{13}$$

As for $\boldsymbol{\beta}$, the original optimization problem can be rewritten as

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n}\left(\sum_{k=1}^{K}\frac{\gamma_{ik}}{2\sigma_k^2}\right)(y_i - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})^2 + \lambda g(\boldsymbol{\beta}), \tag{14}$$

by ignoring the items not depending on $\boldsymbol{\beta}$. Equation (14) implies that the estimate of $\boldsymbol{\beta}$ can be obtained by minimizing the weighted RSS plus the penalty assigned on $\boldsymbol{\beta}$, where the weight for the ith observation is $\tilde{w}_i = \sqrt{\sum_{k=1}^{K}\gamma_{ik}/(2\sigma_k^2)}$. In next subsection, we will discuss how to solve this issue effectively. Although the penalty function $g(\boldsymbol{\beta})$ has many choices, we mainly discuss the situation with $g(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_1$ and abbreviate the novel method as MoG-Lasso whose workflow is summarized in Algorithm 1. We initialize $\boldsymbol{\beta}$ as a zero vector. As for Γ, it is initialized by some random numbers from $U(0, 1)$ (i.e. the uniform distribution on the interval (0,1)).

Algorithm 1 MoG-Lasso

Input: $\mathbf{X}, \mathbf{y}, K, \lambda$.
Output: $\boldsymbol{\beta}, \boldsymbol{\Gamma}, \boldsymbol{\Pi}, \boldsymbol{\Sigma}$.
1: Initialize $\boldsymbol{\beta}, \boldsymbol{\Gamma}$.
2: **while** the convergence criterion does not satisfy **do**
3: Update the terms in Equation (13).
4: Compute the weights $\tilde{w}_i (i = 1, \cdots, n)$ and update $\boldsymbol{\beta}$ by solving Equation (14).
5: Evaluate $\boldsymbol{\Gamma} = (\gamma_{ik})_{n\times K}$ by Equation (11).
6: **end while**

2.2. Solving a penalized weighted regression

As for the optimization problem in (14), several sophisticated algorithms such as iterative shrinkage-thresholding algorithm (ISTA [8]), cyclic coordinate descent algorithm (CCD [29]) and alternating direction method of multipliers (ADMM [3]) have been developed to solve it efficiently. Here, we employ ADMM to solve (14) because of its fast convergence. Originally, ADMM is designed to solve the constraint optimization problem

$$\min_{\boldsymbol{x},\boldsymbol{z}} f(\boldsymbol{x}) + g(\boldsymbol{z}), \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \tag{15}$$

where $\boldsymbol{x} \in \mathbb{R}^s, \boldsymbol{z} \in \mathbb{R}^t, \boldsymbol{c} \in \mathbb{R}^m, \boldsymbol{A} \in \mathbb{R}^{m\times s}, \boldsymbol{B} \in \mathbb{R}^{m\times t}$. Therefore, we can write the augmented Lagrangian function as

$$L = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \rho\boldsymbol{u}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}||\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}||_2^2. \tag{16}$$

Here, ρ denotes the penalty parameter and $\boldsymbol{u} \in \mathbb{R}^m$ is a dual vector. Therefore, we can solve Equation (15) by repeatedly updating $\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}$ as

$$\begin{aligned}
\boldsymbol{x}^{k+1} &= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \frac{\rho}{2}||\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z}^k - \boldsymbol{c} + \boldsymbol{u}^k||_2^2, \\
\boldsymbol{z}^{k+1} &= \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + \frac{\rho}{2}||\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{u}^k||_2^2, \\
\boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}.
\end{aligned} \tag{17}$$

To put our problem, Equation (14), into the framework of ADMM, we first rescale the weights by $w_i = n\tilde{w}_i / \sum_{j=1}^n \tilde{w}_j$ for the purpose of numeric stability. And then, let the working predictor and response be $\boldsymbol{a}_i = w_i \boldsymbol{x}_i$ and $b_i = w_i y_i (i = 1, 2, \ldots, n)$, respectively. As a result, the original optimization problem in (14) is equivalent to

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^n (b_i - \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{\beta})^2 + \lambda g(\boldsymbol{z}) \quad \text{s.t. } \boldsymbol{z} = \boldsymbol{\beta}. \tag{18}$$

With the ADMM algorithm, it is easy to attain the update of parameters as

$$\begin{aligned}
\boldsymbol{\beta}^{k+1} &= (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \rho \boldsymbol{I})^{-1}[\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b} + \rho(\boldsymbol{z}^k - \boldsymbol{u}^k)], \\
\boldsymbol{z}^{k+1} &= \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + \frac{\rho}{2}||\boldsymbol{\beta}^{k+1} - \boldsymbol{z} + \boldsymbol{u}^k||_2^2, \\
\boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \boldsymbol{\beta}^{k+1} - \boldsymbol{z}^{k+1}.
\end{aligned} \tag{19}$$

Note that the update of $\boldsymbol{z}$ depends on penalty function $g(\boldsymbol{z})$. For the case of $g(\boldsymbol{z}) = ||\boldsymbol{z}||_1$, the literature [3] has provided its solution as

$$\boldsymbol{z}^{k+1} = \mathcal{S}_{\lambda/\rho}(\boldsymbol{\beta}^{k+1} + \boldsymbol{u}^k), \tag{20}$$

where the soft thresholding function for a scalar is defined by

$$\mathcal{S}_\kappa(z) = \max(0, z - \kappa) - \max(0, -z - \kappa), \tag{21}$$

For a vector, $\mathcal{S}_\kappa(\boldsymbol{z})$ takes the value $(\mathcal{S}_\kappa(z_1), \ldots, \mathcal{S}_\kappa(z_t))^{\mathrm{T}}$. Algorithm 2 summarizes how to use ADMM to solve a weighted Lasso issue.

Algorithm 2 Weighted Lasso

Input: $\mathbf{X}, \mathbf{y}, \mathbf{w}, \rho, \lambda$.
Output: $\boldsymbol{\beta}$.

1: Initialize $\mathbf{u}, \mathbf{z}$ as zero vectors.
2: Rescale weight $w_i = n\tilde{w}_i / \sum_{j=1}^n \tilde{w}_j$; construct the working response and design matrix: $b_i = w_i y_i$ and $\mathbf{a}_i = w_i \mathbf{x}_i$.
3: **while** the convergence criterion does not satisfy **do**
4: Update $\boldsymbol{\beta}, \mathbf{z}, \mathbf{u}$ according to Equations (19) and (20).
5: **end while**
6: Let $\boldsymbol{\beta} = \mathbf{z}$.

Remark 2.1: In later experiments, we set $\rho = 1$. Since ρ does not change, it is able to save computational resource by caching the Cholesky factorization of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \rho\boldsymbol{I}$. In this manner, we just need to carry out the back-solves by cached factorization when we update $\boldsymbol{\beta}$. Note that the cost of taking Cholesky factorization is not large even if $p > n$. Because of the fact

$$(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \rho\boldsymbol{I})^{-1} = \boldsymbol{I}/\rho - \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}\boldsymbol{A},$$

the target of Cholesky factorization turns into $\boldsymbol{I} + \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ instead of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \rho\boldsymbol{I}$ when $p > n$ (refer to literature [3] for more details).

2.3. Tuning of hyper-parameter

It is well-known that the penalty parameter λ plays a significant role in sparse regression to achieve a good trade-off between prediction accuracy and sparsity. Here, we employ cross validation plus BIC (Bayesian information criterion) to determine the optimal λ^* over some candidate values of λ. Given a penalty parameter λ, BIC is defined as

$$\mathrm{BIC}(\lambda) = \mathrm{RSS}_\lambda + |\mathcal{M}_\lambda| \log(n),$$

in which $\mathrm{RSS}_\lambda = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ and $\hat{y}_i$ denotes the response values predicted by the model $\mathcal{M}_\lambda$. And $|\mathcal{M}_\lambda|$ stands for the number of non-zero coefficients in the model determined by λ. For MoG-Lasso, we followed the strategy proposed in [12] to first choose some sensible candidate values of λ. Regarding Lasso, it has been proved that all the regression coefficients will be zero if λ exceeds

$$\lambda_{\max} = \max_{1 \le j \le p} \left\{ \frac{|\mathbf{X}_j^{\mathrm{T}}\mathbf{y}|}{\mathbf{X}_j^{\mathrm{T}}\mathbf{X}_j} \right\}. \tag{22}$$

Then, Friedman *et al.* [12] suggested to set $\lambda_{\min} = 0.001\lambda_{\max}$ and take a set of M (in general, $M = 100$) different values — equally spaced on the logarithmic scale — between $\lambda_{\min}$ and $\lambda_{\max}$. In the case of MoG-Lasso, we find that Equation (22) often leads to a non-zero solution because of the complex noise distribution. With some preliminary experiments, we empirically set $\lambda_{\max}^{\mathrm{MoG}} = 10\lambda_{\max}$ and $\lambda_{\min}^{\mathrm{MoG}} = 10^{-5}\lambda_{\max}^{\mathrm{MoG}}$ in later simulations. Subsequently, 100 candidate values of λ are taken from $[\lambda_{\min}^{\mathrm{MoG}}, \lambda_{\max}^{\mathrm{MoG}}]$ in the manner similar to that used for Lasso.

3. Numerical experiments

This section devotes to investigating the performance of the proposed model, MoG-Lasso. All the experiments were conducted with R software and run on a computer with Intel Core CPU 3.60 GHz, 8.00 GB RAM and Windows 10(64-bit) system. In the following experiments, we compared MoG-Lasso with Lasso [28], LAD-Lasso [33] and Huber–Lasso [36]. In what follows, we abbreviated these algorithms as MoG, LAD and Huber, respectively. We utilized R package `glmnet` [12][1] to implement Lasso, `hqreg`[2] to implement LAD and Huber. The 10-fold cross validation was adopted to choose the optimal penalty parameter λ for all considered methods.

In subsequent simulations, all data were generated by the linear model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ if not stated otherwise. In each example, the true coefficient vector $\boldsymbol{\beta}$ and the distribution of noise $\boldsymbol{\varepsilon}$ were set differently to study the performance of each method from various aspects.

3.1. Performance metrics

In order to assess the prediction performance of each method, we employed the *relative model error* (*RME*) defined as $\text{RME} = E(\boldsymbol{y} - \hat{\boldsymbol{y}})^2/\text{Var}(\boldsymbol{y})$. In particular, we first generated a test set consisting of $n_{\text{ts}} = 10^5$ observations. Then, we trained each model and tuned its related parameters. Subsequently, the RME was estimated as $\text{RME} = \sum_{i=1}^{n_{\text{ts}}}(y_i - \hat{y}_i)^2 / \sum_{i=1}^{n_{\text{ts}}}(y_i - \bar{y})^2$, where $\hat{y}_i = \boldsymbol{x}_i^{\text{T}}\hat{\boldsymbol{\beta}}$ and $\bar{y} = (1/n_{ts})\sum_{i=1}^{n_{\text{ts}}} y_i$.

To assess *variable selection accuracy* of each model, we used F_1 score defined as

$$\text{F}_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{23}$$

with

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{24}$$

where TP, FP and FN denotes the number of true positives, false positives and false negatives, respectively. Besides F_1 score, we also computed the *mean parameter bias* for the regression coefficients as

$$\text{BIAS} = \frac{1}{p}\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j)^2. \tag{25}$$

In the following simulations, all metrics were averaged over 100 independent trails. It is worth mentioning that an algorithm may lead to a null model (that is, none variables are identified as important) if data is corrupted by severe noise. Thereafter, F_1 is not available because the denominator is zero. In the following experiments, we directly set $\text{F}_1 = 0$ if an algorithm leads to a null model.

3.2. Example 1: Effect of the number of Gaussian components

In the above discussions, we provide an algorithm to solve MoG-Lasso with K Gaussian noise components. Obviously, MoG-Lasso degenerates into Lasso if $K = 1$ and loses the robustness, while too large K makes MoG-Lasso violate Occam Razor's principle. Hence, it is still a question about how to determine K properly. This example is designed to study how the performance of MoG-Lasso varies with different values of K.

We sampled $x_{ij} \sim \mathcal{N}(0, 1)$ and $\varepsilon \sim t(1, 0)$ (i.e. a Student's t distribution with degrees of freedom 1 and location parameter 0) and took $n = 50, p = 100, \boldsymbol{\beta} = (\mathbf{2}_5, \mathbf{0}_{95})^{\text{T}}$, namely, there were only 5 important covariates. We considered MoG-Lasso with $K = 2, 3, \ldots, 8$. For each value of K, we conducted 100 runs of experiments and recorded F_1 score, BIAS, RME as well as the running time (in terms of seconds) of MoG-Lasso. Figure 1(a) depicts

Table 1. The results of experiments in Section 3.2.

K	F_1	BIAS	RME	Time	F_1	BIAS	RME	Time	F_1	BIAS	RME	Time
	$t(1,0)$				$\mathcal{N}(0,4^2)$				$\frac{1}{2}(\mathcal{N}(0,4^2)+\mathcal{N}(0,2^2))$			
2	0.7065	0.0982	0.4926	**7.6280**	*0.6719*	**0.0293**	0.2785	**22.4102**	0.8239	*0.0140*	*0.1328*	**28.7562**
3	0.6676	0.1126	0.6372	*13.0626*	0.6424	0.0308	0.2810	*22.5107*	0.8154	0.0153	0.1428	*34.5847*
4	0.6760	**0.0874**	**0.4225**	14.6316	0.6402	0.0307	**0.2756**	25.9485	*0.8550*	0.0142	0.1346	36.0391
5	0.6990	*0.0954*	*0.4719*	14.6332	**0.6796**	*0.0298*	*0.2775*	25.7805	0.8244	0.0160	0.1566	36.5603
6	0.7042	0.1052	0.5103	13.8138	0.6544	0.0307	0.2857	28.2724	0.8180	0.0160	0.1458	34.7237
7	*0.7073*	0.1116	0.5150	13.4304	0.6311	0.0336	0.3061	27.4154	**0.8565**	**0.0138**	**0.1307**	36.5441
8	**0.7101**	0.1108	0.5270	13.5149	0.6287	0.0351	0.3258	26.5557	0.8219	0.0169	0.1555	39.1516
	$\frac{1}{3}(\mathcal{N}(0,4^2)+\mathcal{N}(0,3^2)+\mathcal{N}(0,2^2))$				$\frac{1}{4}(\mathcal{N}(0,4^2)+\mathcal{N}(0,3.5^2)$ $+\mathcal{N}(0,3^2)+\mathcal{N}(0,2^2))$				$\frac{1}{5}(\mathcal{N}(0,4^2)+\mathcal{N}(0,3.5^2)+$ $\mathcal{N}(0,3^2)+\mathcal{N}(0,2.5^2)+\mathcal{N}(0,2^2))$			
2	0.8242	0.0155	0.1534	**27.2537**	0.8248	**0.0139**	**0.1553**	**22.9002**	0.8327	0.0140	0.1531	**24.9092**
3	0.8559	**0.0121**	**0.1252**	*29.2955*	0.8052	0.0177	0.1869	28.2966	0.8295	0.0142	0.1550	28.0787
4	*0.8591*	0.0130	0.1348	33.7428	**0.8270**	0.0146	*0.1560*	30.5376	0.8181	0.0167	0.1793	*25.7447*
5	0.8488	*0.0126*	*0.1256*	30.2844	*0.8269*	0.0147	0.1576	31.1247	0.8449	*0.0126*	*0.1344*	29.7101
6	0.8390	0.0137	0.1373	31.6817	0.8115	0.0151	0.1602	28.9054	0.8357	0.0156	0.1635	28.7298
7	0.8350	0.0140	0.1411	30.9531	0.8182	*0.0145*	0.1572	32.0782	*0.8541*	*0.0126*	0.1348	31.2402
8	**0.8629**	0.0131	0.1335	35.9112	0.7967	0.0156	0.1673	32.5508	**0.8567**	**0.0122**	**0.1298**	28.6451

Note: The results for the best and the second best methods are highlighted in boldface and italics, respectively.

the error bar plots of the obtained results. In the meantime, we also listed the mean values of the considered metrics in Table 1.

It is shown that, as the growth of K, MoG tends to be slow. Furthermore, although $K=8$ sometimes reaches the highest F_1 value, but its BIAS and RME are higher than those of $K=4$. It seems that $K=4$ strikes a good balance between variable selection and prediction, since there is no need to employ too many Gaussian components to fit the $t(1,0)$ distribution. But if we take computational cost into account, $K=2$ seems to be a better choice because it is the fastest one and has good performance in terms of both variable selection and prediction accuracy. Furthermore, as shown in Figure 1, there is no significant difference between the results for different K values.

Thereafter, we did more experiments with noise from various mixtures of Gaussians. In particular, we set $n=100$ and $p=200$. The noise was simulated from the mixture of K^{true} Gaussian components, where $K^{\text{true}}=1,2,\ldots,5$, while we implemented our method by setting $K=2,3,\ldots,8$. Analogous to the previous experiment, we also carried out 100 runs of experiments for each value of K, and recorded the F_1 score, BIAS, RME and the running time of MoG-Lasso. The obtained results are reported in Table 1 and Figure 1, respectively. Intuitively, the optimal K should be K^{true} since the real noise is from the mixture of K^{true} Gaussians. However, it is shown, in Table 1 that, on one hand, the best performance is not always achieved by the model with $K=K^{\text{true}}$; on the other hand, the larger K does not necessarily lead to better performance. Furthermore, Figure 1 also shows that the performance of different K values is similar. Based on the fact that $K=2$ is fastest and has competitive performance, we recommend to setting $K=2$ in MoG-Lasso.

As a matter of fact, the above conclusions are not surprising. Consider one simple case which assumes the noise being sampled from a Gaussian distribution with an extremely large variance. There is no doubt that Lasso performs badly in this case, even though Lasso indeed obeys the assumption (that is, Lasso's loss function is exactly compatible with the true noise distribution). In the opposite, MoG-Lasso with $K=2$ can successfully

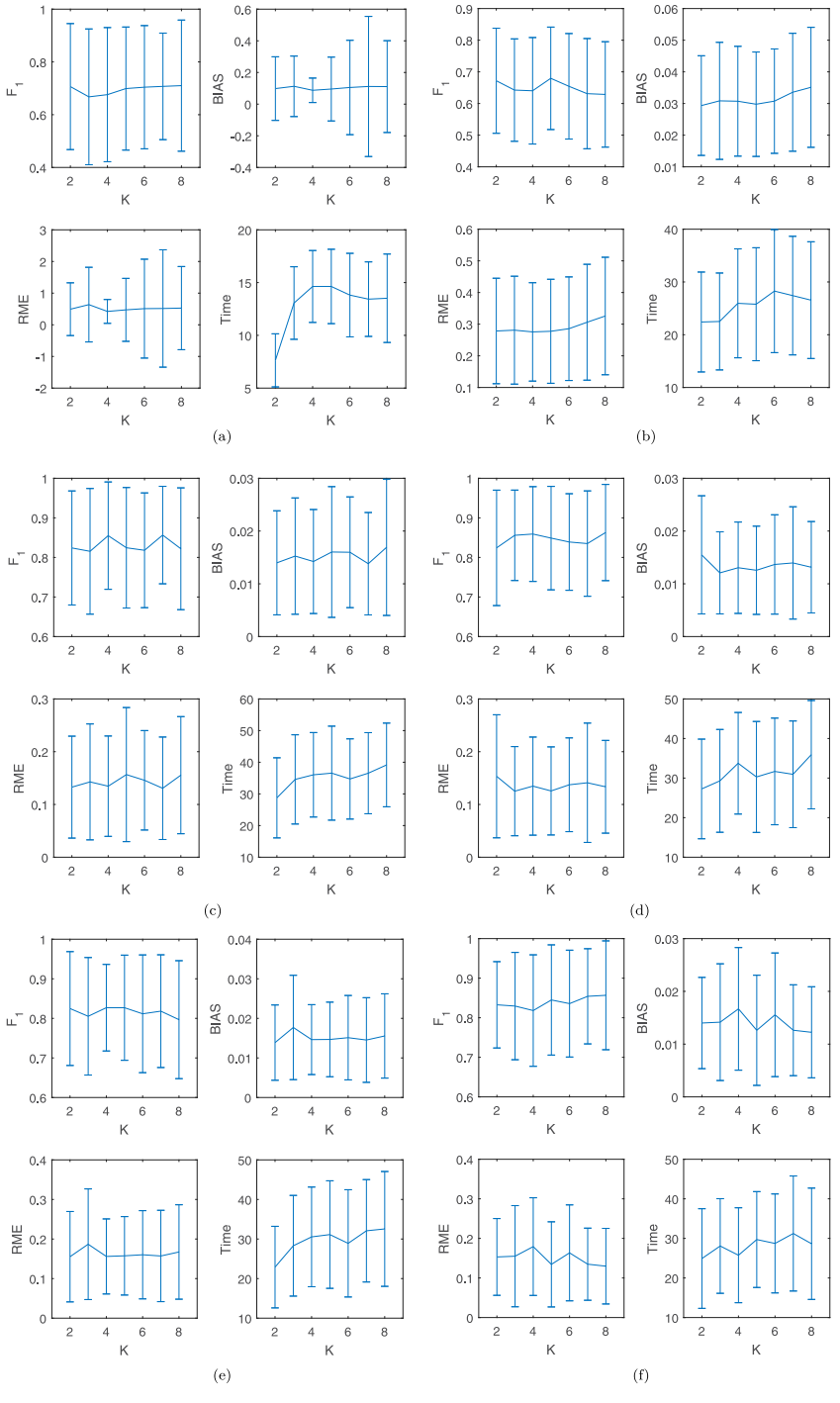

Figure 1. The error bar plots of F_1, BIAS, RME and time over 100 random experiments in Section 3.2. (a) $t(1,0)$. (b) $\mathcal{N}(0,4^2)$. (c) $\frac{1}{2}(\mathcal{N}(0,4^2)+\mathcal{N}(0,2^2))$. (d) $\frac{1}{3}(\mathcal{N}(0,4^2)+\mathcal{N}(0,3^2)+\mathcal{N}(0,2^2))$. (e) $\frac{1}{4}(\mathcal{N}(0,4^2)+\mathcal{N}(0,3.5^2)+\mathcal{N}(0,3^2)+\mathcal{N}(0,2^2))$. (f) $\frac{1}{5}(\mathcal{N}(0,4^2)+\mathcal{N}(0,3.5^2)+\mathcal{N}(0,3^2)+\mathcal{N}(0,2.5^2)+\mathcal{N}(0,2^2))$.

handle this problem since two Gaussian components account for the outliers and inliers, respectively.

3.3. Example 2: Simulations with various kinds of noise

In this example, we did some experiments to compare MoG-Lasso with some other algorithms. The design matrix $\boldsymbol{X}$ was drawn from a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ where $\Sigma_{ij} = 0.5^{|i-j|}$. Here, we took $n = 50, p = 100$ and the true coefficient vector $\boldsymbol{\beta} = (\mathbf{2}_5, \mathbf{0}_{95})^{\mathrm{T}}$. As shown in Table 2, six kinds of noise were considered, including four Gaussian distributions $\mathcal{N}(0, \sigma^2)$ where the standard deviation σ takes values 0.5,2,5,7, respectively, a Student's t distribution $t(1, 0)$ and a mixture of t distributions, i.e. $0.5t(1, -2) + 0.5t(1, 2)$. Over 100 independent trails, the average signal-to-noise ratio $\mathrm{Var}(\boldsymbol{X\beta})/\mathrm{Var}(\boldsymbol{\varepsilon})$ of the first four cases were 184.55, 11.89, 1.85, 0.97, respectively. Table 2 summarizes the evaluation metrics for each method.

It can be seen in Table 2 that MoG always outperforms the other methods in terms of all metrics. It is interesting that in the Gaussian noise cases with $\sigma = 0.5, 2, 3$, Huber and LAD unexpectedly do poorly in terms of variable selection and prediction, compared with Lasso. This may be caused by the fact that the loss function of Lasso is L_2 norm which coincides with the real noise distribution. To attain more insights on the behavior of every method, Figure 2 shows some details for a random experiment with $\sigma = 5$. The top panel of Figure 2 provides the trace plots of regression coefficients, where the dashed lines indicate the model with the optimal λ being tuned by 10-fold cross validation. It is shown that, on one hand, all methods correctly identify unimportant covariates; on the other hand, LAD misses two important variables, while the others miss one. Furthermore, we observe that the bias of $\boldsymbol{\beta}$ in MoG is smaller than others. The middle panel of Figure 2 shows the scatter plots of true noise versus residuals over all training samples. Here, the residual of the ith sample is defined as $r_i = y_i - \boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}$. Evidently, the prediction accuracy of a model is higher if the residuals are closer to the true noise. Since the residuals of MoG almost coincides with noise, there is no doubt that MoG achieves the best prediction among all methods. At last, the histograms of residuals are offered in the bottom panel of Figure 2. The residual distributions of MoG and Lasso looks like Gaussians. However, those of Huber and LAD seem to be strange. Moreover, Table 2 manifests that Lasso is outperformed by LAD and

Table 2. The results of experiments in Section 3.3.

	F1	BIAS	RME	F1	BIAS	RME	F1	BIAS	RME
	$\sigma = 0.5$			$\sigma = 2$			$\sigma = 5$		
MoG	**1.0000**	**0.0009**	**0.0023**	**0.9613**	**0.0154**	**0.0331**	**0.7189**	**0.0971**	**0.2219**
Lasso	*0.8694*	0.0014	0.0041	*0.8551*	*0.0200*	0.0600	*0.7143*	*0.0996*	*0.3407*
Huber	0.8571	*0.0013*	*0.0036*	0.7547	0.0242	*0.0592*	0.6239	0.1138	0.3680
LAD	0.7011	0.0019	0.0050	0.7080	0.0281	0.0727	0.5714	0.1183	0.3598
	$\sigma = 7$			$t(1, 0)$			$0.5t(1, -2) + 0.5t(1, 2)$		
MoG	**0.5215**	**0.1341**	**0.4055**	**0.7983**	**0.0675**	**0.1640**	**0.7595**	**0.0752**	**0.1773**
Lasso	0.4974	0.1491	0.5701	0.3045	0.1558	0.7343	0.3007	0.1596	0.7472
Huber	*0.4946*	0.1545	0.5160	*0.6963*	*0.0881*	*0.3753*	*0.6412*	*0.1050*	*0.4315*
LAD	0.5006	*0.1481*	*0.4777*	0.6466	0.0936	0.4076	0.5072	0.1224	0.5042

Note: The results for the best and the second best methods are highlighted in boldface and italics, respectively.

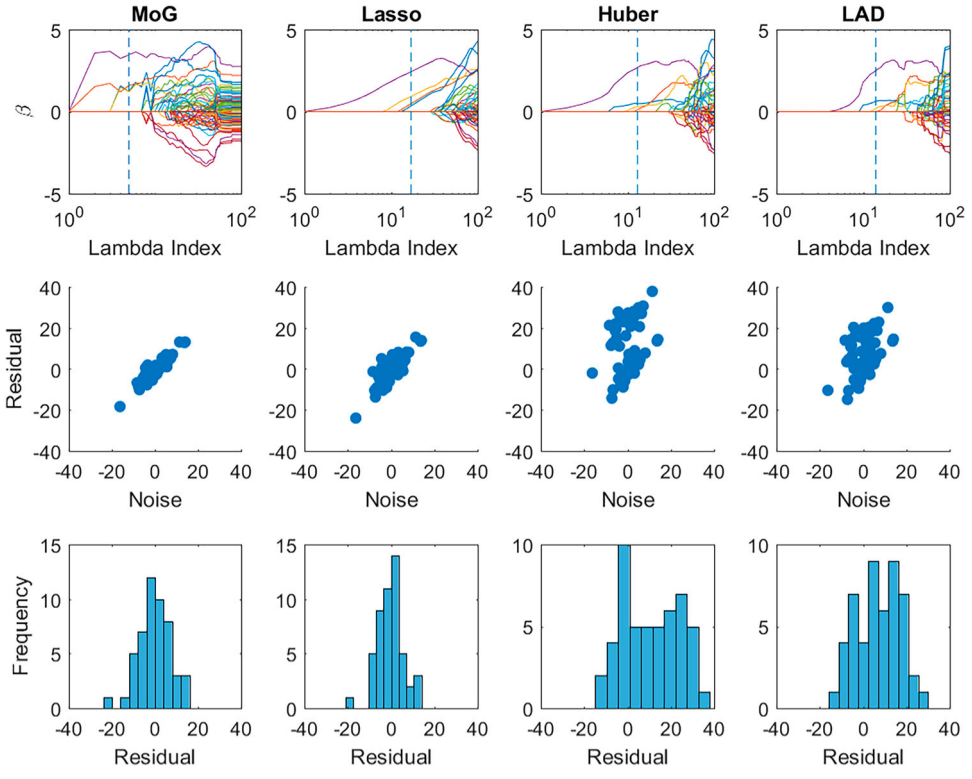

Figure 2. Top panel: The trace plots of regression coefficient. Middle panel: The scatter plots of true noise versus residuals. Bottom panel: The histograms of residuals. The subplots from left to right correspond to MoG-Lasso, original Lasso, Huber-Lasso and LAD-Lasso, respectively.

Huber when it comes to the three situations with heavy-tailed noise (i.e. $\mathcal{N}(0, 7^2)$, $t(1, 0)$ and $0.5t(1, -2) + 0.5t(1, 2)$). In conclusion, Huber and LAD are suitable for heavy-tailed noise rather than Gaussian noise. In contrast, MoG is able to handle both kinds of noise very well.

3.4. Example 3: Diets simulation

This example was modified from [13]. There were two groups of diets with $n/2$ subjects in each group. Let $x_{i1} = \mathbb{I}(i > n/2) - \mathbb{I}(i \leq n/2)$ indicate the group of ith subject ($i = 1, 2, \ldots, n$). Subsequently, for $j = 2, \ldots, 101$, we simulated x_{ij} as $x_{ij} = u_{ij} + x_{i1}v_j$, where u_{ij} were sampled from $U(0, 1)$, $v_2, \ldots, v_{76}$ were sampled from $U(0.25, 0.75)$ and $v_{77}, \ldots, v_{101}$ were zeros. The first 75% covariates are highly correlated ($|\rho| \approx 0.8$), while the remaining covariates are weakly correlated ($|\rho| \approx 0.4$). In summary, there were 101 covariates and the true coefficient vector was taken as $\boldsymbol{\beta} = (4.5, 3, -3 - 3, \mathbf{0}_{96}, 3)^{\mathrm{T}}$. In our experiments, we considered $n = 60$,100 and 200, respectively. Finally, we added noise sampled from a Laplace distribution with mean 2.

The results listed in Table 3 demonstrate that the correlation between covariates dramatically reduce the variable selection accuracy and prediction accuracy of each method. The behavior of Lasso, Huber and LAD in this example are remarkably inferior to that of MoG,

Table 3. The results of experiments in Section 3.4.

	$n = 60$			$n = 100$			$n = 200$		
	F_1	BIAS	RME	F_1	BIAS	RME	F_1	BIAS	RME
MoG	**0.3500**	**0.3726**	**0.3017**	**0.4360**	**0.3395**	**0.2300**	**0.7852**	**0.1487**	**0.1146**
Lasso	0.1991	0.5295	0.8640	0.2772	0.4730	*0.6829*	*0.3202*	0.4263	1.4222
Huber	0.2224	*0.5250*	1.5225	*0.2876*	*0.4671*	1.2358	0.2948	*0.4059*	*0.9174*
LAD	*0.2517*	0.5446	*0.8263*	0.2566	0.4834	0.9974	0.2794	0.4119	1.3958

Note: The results for the best and the second best methods are highlighted in boldface and italics, respectively.

no matter whether the sample size n is small or large. It implies that Huber and LAD can hardly recover the heavy-tailed noise if the correlation between covariates is complicated.

3.5. Example 4: Weak signal simulation

This example was designed to study the ability of each method to recover weak signals. Here, a weak signal means that the coefficient of a truly important variable approaches to zero or is much smaller than others. In this example, we conducted two groups of simulations: (1) $n = 100$, $p = 200$ and there were 5 important variables (that is, the first five ones) taking values from 0.5 to 2 with equal increments; (2) $n = 200$, $p = 512$ and there were 10 important variables (that is, the first ten ones) taking values from 0.5 to 2 with equal increments. In both groups, the $t(1, 0)$ noise was employed and $x_{ij} \sim \mathcal{N}(0, 1)$. Note that several regression coefficients are small, part of which are very likely to be omitted.

The results in Table 4 demonstrate that the F_1 scores are significantly lower than those in Example 2. Specifically, the F_1 score of Lasso almost approaches to 0 and Huber, to some extent, achieves slightly higher F_1 socre. By contrast, MoG in this example still outperforms the other counterparts, and LAD is the second best method. In terms of RME, however, LAD is observed to perform better than MoG when $n = 100$, $p = 200$. After checking the results of LAD and MoG over 100 experiments, we found that both LAD and MoG fail on one experiment. For the other experiments, MoG attains lower RME than LAD in most cases. In the experiment that both methods fail, LAD generates a null model. In other words, the coefficients of all variables are zero. Thus, the evaluation metrics of LAD on this experiment are $F_1 = 0$, BIAS $= 0.0461$, RME $= 1$. In contrast, MoG leads to a very dense model, where only 7 unimportant variables coefficients are excluded from the model. Therefore, the regression coefficients are badly estimated. The evaluation metrics of MoG are $F_1 = 0.0505$, BIAS $= 0.1357$, RME $= 3.1304$. In summary, the performance of MoG is greatly discounted by this failure.

Table 4. The results of experiments in Section 3.5.

	$n = 100, p = 200$			$n = 200, p = 512$		
	F_1	BIAS	RME	F_1	BIAS	RME
MoG	**0.6427**	**0.0235**	*0.5358*	**0.6010**	**0.0172**	**0.4645**
Lasso	0.1109	0.0425	0.9159	0.0084	0.0348	0.9950
Huber	0.4612	0.0286	0.5957	0.4313	0.0229	0.6539
LAD	*0.5622*	*0.0246*	**0.4998**	*0.4458*	*0.0227*	*0.6483*

Note: The results for the best and the second best methods are highlighted in boldface and italics, respectively.

3.6. Real data analysis

In this section, we aim at studying the performance of MoG and the other sparse regression methods on two real-world data sets, namely, triazines[3] [19] and GSE5680[4] [27]. The data set triazines is a pharmaceutical development dataset collecting 186 compounds testing inhibition of mouse tumor and 60 features. GSE5680 is the microarray expression data of 31042 probe sets from 120 rat samples. For GSE5680, the interest of researchers is to find the genes whose expression is correlated with that of gene TRIM32, since this gene causes Bardet–Biedl syndrome [7]. Since the probe from TRIM32 is 1389163_at, it was thus taken as the response and the other probes were treated as predictors. Similar to [15], we selected 5000 probes with the greatest variance and then computed the correlation coefficients between them and TRIM32. Thereafter, the 1000 probes with the strongest correlation coefficient were selected as the final covariates. Therefore, our dataset is with $n = 120$, $p = 1000$.

To address the performance of each method, we ran 10 times the 10-fold cross validation to estimate the RME and recorded sparsity (i.e. the number of excluded variables). Table 5 summarizes the means and standard deviations of RME and sparsity for each method. On the triazines dataset, MoG is observed to have the best prediction ability as well as the strongest interpretation power, while the performance of Huber and LAD is inferior to Lasso. Moreover, we also plotted the boxplot of the 100 RME and sparsity values of each method in Figure 3 in order to get more insights about their relative performance. The Huber and LAD on triazines dataset tend to select more covariates, while Figure 3 manifests that they lead to lower prediction accuracy.

As for GSE5680 dataset, on the one hand, MoG still achieves the lowest prediction error and the RMEs of Huber and LAD are slightly higher than that of Lasso. On the other hand, Lasso and MoG lead to the sparsest and densest models, respectively. From Table 5, we

Table 5. The results of real-world datasets.

	Triazines		GSE5680	
	RME	Sparsity	RME	Sparsity
MoG	**7.5843**± 4.4592	**56.68**± 0.66	**277.3344**± 224.6330	963.95± 4.01
Lasso	*18.5676*± 6.4509	*55.35*± 3.28	*4051.4052*± 1864.5362	*985.26*± 4.86
Huber	26.9137± 9.6242	53.99± 2.61	4055.0612± 2594.6046	**985.69**± 2.61
LAD	19.1843± 8.4706	53.83± 1.65	5029.1449± 2362.8747	984.84± 2.82

Note: The results for the best and the second best methods are highlighted in boldface and italics, respectively.

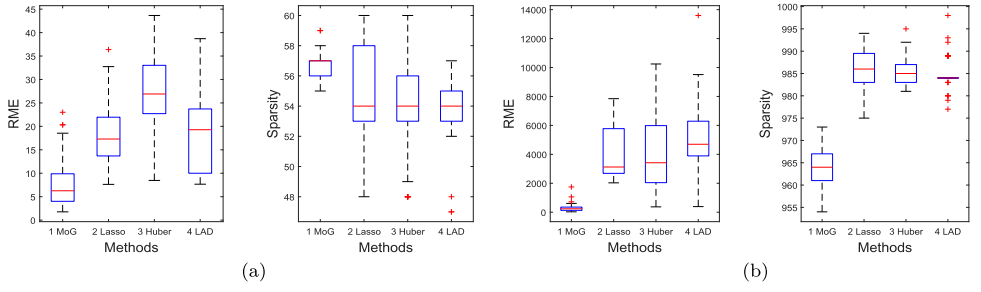

Figure 3. The boxplot of RME and sparsity over 100 implements on real datasets.

know that Lasso, Huber and LAD select 15 probes on average, while MoG includes 36 probes. Though Lasso, Huber and LAD are much sparser than MoG, they cannot exactly predict the expression of TRIM32. In conclusion, MoG provides more reliable results on this dataset.

4. Discussions

Even though we focus on the L_1 penalty in previous discussions, the framework of MoG-Lasso can be applied to other sparse penalty functions and the only difference is Equation (20). For example, if we employ the L_0 or $L_{1/2}$ penalty, the updates of $\boldsymbol{z}$ will be $\mathcal{D}_{\lambda/\rho}(\boldsymbol{\beta}^{k+1} + \boldsymbol{u}^k)$ and $\mathcal{F}_{\lambda/\rho}(\boldsymbol{\beta}^{k+1} + \boldsymbol{u}^k)$, respectively, where the hard and half [35] thresholding functions are defined by

$$\mathcal{D}_\kappa(z) = z\mathbb{I}(|z| > \kappa), \tag{26}$$

$$\mathcal{F}_\kappa(z) = \frac{2z}{3}\left[1 + \cos\left(\frac{2(\pi - \psi)}{3}\right)\right]\mathbb{I}\left(|z| > \frac{3}{4}\kappa^{2/3}\right), \tag{27}$$

where $\psi = \arccos(\kappa/8(|z|/3)^{-3/2})$.

5. Conclusions

Because the noise contained in real-world data is complex, the models established with some specific noise distributions (such as Gaussian, Laplace and etc.) may perform not very well. They are also not robust enough to outliers. In this paper, we had proposed a robust sparse regression model MoG-Lasso without any assumption about the noise distribution. The core idea is to directly model noise by an MoG, where the MoG parameters can be automatically learned from data. To enhance its ability to identify a sparse model, the L_1 penalty was included as a part of the loss function of MoG-Lasso. We also presented an efficient algorithm to estimate its associated parameters via the EM and ADMM algorithms. By conducting experiments with simulated and real data which consist of complex noise, the results show that MoG-Lasso performs better than several other existing counterparts.

Notes

1. https://CRAN.R-project.org/package = glmnet
2. https://CRAN.R-project.org/package = hqreg
3. http://www.dcc.fc.up.pt/ ~ ltorgo/Regression/DataSets.html
4. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc = GSE5680

Acknowledgements

The authors would like to thank the editor and the referees for their valuable comments which greatly helped to improve the paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Natural Science Foundation of China [grant number 11671317].

References

[1] A. Alfons and S. Gelper, *Sparse least trimmed squares regression for analyzing high-dimensional large data sets*, Ann. Appl. Stat. 7 (2013), pp. 226–248.
[2] G. Bassett Jr and R. Koenker, *Asymptotic theory of least absolute error regression*, J. Amer. Statist. Assoc. 73 (1978), pp. 618–622.
[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn. 3 (2011), pp. 1–122.
[4] B.S. Cade and B.R. Noon, *A gentle introduction to quantile regression for ecologists*, Front. Ecol. Environ. 1 (2003), pp. 412–420.
[5] E. Candes and T. Tao, *The Dantzig selector: Statistical estimation when p is much larger than n*, Ann. Stat. 35 (2007), pp. 2313–2351.
[6] Y. Chen, C. Caramanis, and S. Mannor, *Robust sparse regression under adversarial corruption*, International Conference on Machine Learning, Atlanta, USA. PMLR, 2013, pp. 774–782.
[7] A.P. Chiang, J.S. Beck, H.J. Yen, M.K. Tayeh, T.E. Scheetz, R.E. Swiderski, D.Y. Nishimura, T.A. Braun, K.Y. Kim, and J. Huang, *Homozygosity mapping with SNP arrays identifies TRIM32, an E3 ubiquitin ligase, as a Bardet-Biedl syndrome gene (BBS11)*, Proc. Nat. Acad. Sci. USA 103 (2006), pp. 6287–6292.
[8] I. Daubechies, M. Defrise, and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Commun. Pure. Appl. Math. 57 (2004), pp. 1413–1457.
[9] A.P. Dempster, N.M. Laird, and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, J. R. Stat. Soc. 39 (1977), pp. 1–38.
[10] N. El Karoui, D. Bean, P.J. Bickel, C. Lim, and B. Yu, *On robust regression with high-dimensional predictors*, Proc. Nat. Acad. Sci. USA 110 (2013), pp. 14557–14562.
[11] A.C. Faul and M.E. Tipping, *A variational approach to robust regression*, International Conference on Artificial Neural Networks, Vienna, Austria. Springer, 2001, pp. 95–102.
[12] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, J. Stat. Softw. 33 (2010), pp. 1–22.
[13] T.P. Garcia, S. Müller, R.J. Carroll, T.N. Dunn, A.P. Thomas, S.H. Adams, S.D. Pillai, and R.L. Walzem, *Structured variable selection with q-values*, Biostatistics 14 (2013), pp. 695–707.
[14] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*, Chapman & Hall/CRC, Baca Raton, 2015.
[15] J. Huang, S. Ma, and C.H. Zhang, *Adaptive Lasso for sparse high-dimensional regression*, Statist. Sinica. 18 (2006), pp. 1603–1618.
[16] P.J. Huber, *Robust estimation of a location parameter*, Ann. Math. Stat. 35 (1964), pp. 73–101.
[17] P.J. Huber, *Robust regression: Asymptotics, conjectures and Monte Carlo*, Ann. Stat. 1 (1973), pp. 799–821.
[18] J.A. Khan, S.V. Aelst, and R.H. Zamar, *Robust linear model selection based on least angle regression*, J. Amer. Statist. Assoc. 102 (2007), pp. 1289–1299.
[19] R.D. King, J.D. Hirst, and M.J.E. Sternberg, *Comparison of artificial intelligence methods for modeling pharmaceutical qsars*, Appl. Artif. Intell. 9 (1995), pp. 213–233.
[20] K. Knight, *Limiting distributions for L_1 regression estimators under general conditions*, Ann. Stat. 26 (1998), pp. 755–770.
[21] V. Maz'ya and G. Schmidt, *On approximate approximations using Gaussian kernels*, IMA J. Numer. Anal. 16 (1996), pp. 13–29.
[22] D. Meng and F.D.L. Torre, *Robust matrix factorization with unknown noise*, IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, IEEE Computer Society, 2013, pp. 1337–1344.

[23] H. Park, *Outlier-resistant high-dimensional regression modelling based on distribution-free outlier detection and tuning parameter selection*, J. Stat. Comput. Simul. 87 (2017), pp. 1799–1812.
[24] C.R. Rao, H. Toutenburg, Shalabh, and C. Heumann, *Linear Models and Generalizations: Least Squares and Alternatives*, 3rd ed., Springer-Verlag, New York, 2008.
[25] S. Rosset and J. Zhu, *Piecewise linear regularized solution paths*, Ann. Stat. 35 (2007), pp. 1012–1030.
[26] P. Rousseeuw, *Least median of squares regression*, J. Amer. Statist. Assoc. 79 (1984), pp. 871–880.
[27] T.E. Scheetz, K.Y. Kim, R.E. Swiderski, A.R. Philp, T.A. Braun, K.L. Knudtson, A.M. Dorrance, G.F. Dibona, J. Huang, and T.L. Casavant, *Regulation of gene expression in the mammalian eye and its relevance to eye disease*, Proc. Nat. Acad. Sci. USA 103 (2006), pp. 14429–14434.
[28] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, J. R. Stat. Soc. 58 (1996), pp. 267–288.
[29] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Math. Program. 117 (2009), pp. 387–423.
[30] M.J. Wagner, T.H. Kim, J. Savall, M.J. Schnitzer, and L. Luo, *Cerebellar granule cells encode the expectation of reward*, Nature 544 (2017), pp. 96–100.
[31] L. Wang, *The L_1 penalized LAD estimator for high dimensional linear regression*, J. Multivar. Anal. 120 (2013), pp. 135–151.
[32] T. Wang, Q. Li, B. Chen, and Z. Li, *Multiple outliers detection in sparse high-dimensional regression*, J. Stat. Comput. Simul. 88 (2018), pp. 89–107.
[33] H. Wang, G. Li, and G. Jiang, *Robust regression shrinkage and consistent variable selection through the LAD-Lasso*, J. Bus. Econ. Stat. 25 (2007), pp. 347–355.
[34] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma, *Robust face recognition via sparse representation*, IEEE. Trans. Pattern. Anal. Mach. Intell. 31 (2009), pp. 210–227.
[35] Z. Xu, X. Chang, F. Xu, and H. Zhang, *$L_{1/2}$ regularization: A thresholding representation theory and a fast solver*, IEEE. Trans. Neural. Netw. Learn. Syst. 23 (2012), pp. 1013–1027.
[36] C. Yi and J. Huang, *Semismooth Newton coordinate descent algorithm for elastic-net penalized huber loss regression and quantile regression*, J. Comput. Graph. Stat. 26 (2017), pp. 547–557.

Appendix

Here, we provide some details on the derivation of Equations (12) and (13). Note that the Q-function is the expectation of the log-likelihood function with regard to latent variables. In MoG-Lasso, the latent variables are l_{ik}s ($i = 1, \ldots, n; k = 1, \ldots, K$) and the expectation of l_{ik} is given by Equation (11). Thus, the Q-function is

$$\begin{aligned} Q(\boldsymbol{\Omega}) &= E_{l_{ik}|y_i, i=1,\ldots,n, k=1,\ldots,K}\{\ell(\boldsymbol{\Omega}) + \lambda g(\boldsymbol{\beta})\} \\ &= E_{l_{ik}|y_i, i=1,\ldots,n, k=1,\ldots,K}\left\{\sum_{i=1}^{n}\sum_{k=1}^{K}\log[\pi_k p(y_i|\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}, \sigma_k^2)]^{l_{ik}} + \lambda g(\boldsymbol{\beta})\right\} \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K}\gamma_{ik}\log[\pi_k p(y_i|\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}, \sigma_k^2)] + \lambda g(\boldsymbol{\beta}) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K}\gamma_{ik}\left[\log\pi_k - \log\sigma_k^2 - \frac{(y_i - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})^2}{2\sigma_k^2}\right] + \lambda g(\boldsymbol{\beta}). \end{aligned} \tag{A1}$$

Then, $\boldsymbol{\Pi}$ and $\boldsymbol{\Sigma}$ can be attained by maximizing Q-function. By computing the gradient $\partial Q/\partial\sigma_k^2$ and letting it be zero, we can get the update of σ_k^2 as

$$\sigma_k^2 = \frac{1}{\sum_{i=1}^{n}\gamma_{ik}}\sum_{i=1}^{n}\gamma_{ik}\left(y_i - \boldsymbol{x_i}^{\mathrm{T}}\boldsymbol{\beta}\right)^2. \tag{A2}$$

As for π_k, we need to solve the constrained optimization problem

$$\max_{\pi_k} \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \log \pi_k, \quad \text{s.t.} \quad \sum_{k=1}^{K} \pi_k = 1. \tag{A3}$$

According to the Lagrangian multiplier method, we can acquire the update of π_k, that is,

$$\pi_k = \frac{\sum_{i=1}^{n} \gamma_{ik}}{n} = \frac{N_k}{n}. \tag{A4}$$