Project For Convex Optimization: Analysis of Lasso

Liu Zhihan[1,*] (PB18051211)
Lin Ziyuan [1,*] (PB18151818)

January 31, 2021

1 Introduction

In statistics, linear regression model is one of the most fundamental setting for parameter estimation. Original linear regression model is defined as :

$$y = X\beta + e, \quad \text{where } E(e) = 0, Cov(e) = \sigma^2 \tag{1}$$

And in the above model, our goal is simple: Use the predictor matrix X and response vector y to predict β. This problem had already been well studied and a mostly prefect solution, **Least-Squares Method(LS)**,[1] was proposed by Adrien-Marie Legendre in 1877.

The idea of LS is to minimize the norm of the residuals e, i.e.min $||y - X\beta||$, then we can derive [2]the LS estimate $\beta^{OLS} = \left(X^T X\right)^{-} X^T y$. We say it is mostly perfect because on the one hand it can be proved to be the only best linear unbiased estimate by Gauss-Markov Theorem, but on the other hand it cannot cope with the situations when the predictor X is multicollinear, meaning that more than two explanatory variables in X are highly linearly related. It is because if we introduce another evaluation standard , which is defined as $MSE(\hat{\theta}) := E||\theta - \hat{\theta}||$, the MSE of LS estimate can be extreme big when one of the singular value of X is close to zero when X is multicollinear(In fact, these two situations are equivalent).

Since a multicollinear predictor matrix is very common in the practice, it is in need of a new estimate to solve the problem. Later Proposed by Andrey Tikhonov, **Ridge Regression**, as a special case of Tikhonsov regularization, provides improved efficiency in parameter estimation problems and solve the problem in exchange for a tolerable amount of bias. The idea of Ridge Regression is to add positive elements to the diagonals of $X^T X$, thereby decreasing its condition number, i.e. $\beta^{\text{Ridge}} = \left(X^T X + N\lambda I\right)^{-1} X^T y$. We can adjust the hyper parameter λ so that we can balance the bias and efficiency of the estimator.

Whereas, there still exists a problem on Ridge Regression: low model interpretation ability. For l_2 penalty will just apply weight decaying but will not force weight to zero, meaning that the regression (or prediction) process still use all the predictor matrix X instead of selecting

[1]To distinguish with other method, we will denote it as Ordinary Least Squares Method, i.e. OLS

[2]The detailed derivation of the closed form solution of Least Square Method and Ridge Regression will be in the Section 2.4

feature, in other word, the feature used by the estimator is not sparse and it is hard for human to directly extract what the model learnt from the regression (or prediction).

In 1996, Robert Tibshirani rediscovered the prior work on l_1 penalty and propose **Least Absolute Shrinkage and Selection Operator(LASSO)** method and a practical algorithm to deal with the undifferentiated point of $g(x) := |x|$ at origin. And by now it is widely used in practice for its good sparse feature structure and fast convergence.

And our main goal in this paper is to answer the following two problems:

Problem 1 *What's the effect of Lasso and when should we adopt it?*

Problem 2 *How can we solve Lasso problem efficiently and effectively?*

2 Theoretical Analysis of Lasso and l_1 penalty

2.1 Basic Form of Lasso

First we consider the basic form of lasso on the linear regression problem1 ,the object of lasso method is to solve a constrained optimization problem

$$\min_{\beta\in\mathbb{R}^p}\left\{\frac{1}{N}\|y-X\beta\|_2^2\right\} \text{ subject to } \|\beta\|_1\le t \tag{2}$$

For simplicity in usage, we consider the Lagrangian form with l_1 penalty

$$\min_{\beta\in\mathbb{R}^p}\left\{\frac{1}{N}\|y-X\beta\|_2^2+\lambda\|\beta\|_1\right\} \tag{3}$$

Here λ is a hyper parameter selected by the specific data set.

2.2 Generalized form of l_1 penalty regularization

Here we discuss the l_1 penalty on any objective function $f(x,y,w)$, we can also solve a constrained optimization problem

$$\min_{w\in\mathbb{R}^p}\{f(X,Y,w)\} \text{ subject to } \|w\|_1\le t \tag{4}$$

For simplicity in usage, we consider the Lagrangian form with l_1 penalty

$$\min_{w\in\mathbb{R}^p}\{f(X,Y,w)+\lambda\|w\|_1\} \tag{5}$$

Here λ is a hyper parameter selected by the specific data set.

2.3 Interpretation of Lasso

To answer the Problem 1, we can interpret the lasso formula (3) in three point of views. And for convenience, we just discuss the generalized form of l_1 penalty and Lasso has the same property.

2.3.1 Geometry Interpretation

The restriction area by L^p norm $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$is different, if $n = 2$, when $p = 1$ it's a square rotated by 90 degrees and when $p = 2$ it's a circle.

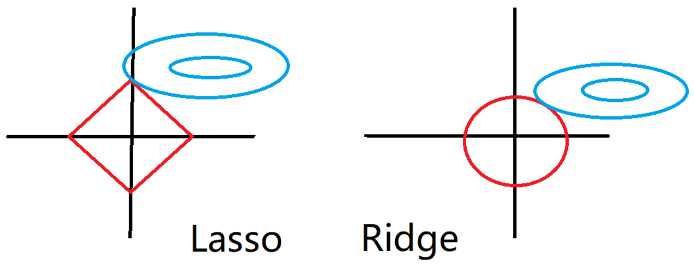

Figure 1: Comparison of Lasso and Ridge from constraint area. Red lines represent Constraint Boundary and blue lines mean the loss curve.

Consider the constrained optimization problem(2), to reduce over-fitting, we adopt regularization by $l_1$norm on the estimator. Plot the loss curve and the constrain area in Figure1, we see that compare with Ridge Regression which use l_2 norm as regularization, our constrain area is sharp, enforcing the optimal point more likely close to the axis while Ridge regression just use weight decay and relative mildly shrink weight to zero which will not force the minor weight to zero.

2.3.2 Bayesian Interpretation

We can also consider from Bayesian view. Laplace distribution is defined as:

$$f(x \mid \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right) = \frac{1}{2b} \begin{cases} \exp\left(-\frac{\mu-x}{b}\right) & \text{if } x < \mu \\ \exp\left(-\frac{x-\mu}{b}\right) & \text{if } x \geq \mu \end{cases}$$

From its plot we see that it is steeper around the mean than Normal distribution.

From Bayesian view, the parameter is a random variable.So if we choose Laplace distribution as the prior distribution of β, then we can deduce the Least squares estimate with l_1 penalty by maximizing the posterior probability:

$$\begin{aligned} \text{From } \arg\max_{\mathbf{w}} L(\mathbf{w}) &= \ln \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\mathbf{y}_i - \mathbf{x}_i\mathbf{w}^\top}{\sigma}\right)^2\right) \cdot \prod_{j=1}^{d} \frac{1}{2b} \exp\left(-\frac{|\mathbf{w}_j|}{b}\right) \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(\mathbf{y}_i - \mathbf{x}_i\mathbf{w}^{\mathrm{T}}\right)^2 - \frac{1}{2\tau^2} \sum_{j=1}^{d} |\mathbf{w}_j| - n \ln \sigma\sqrt{2\pi} - d \ln \tau\sqrt{2\pi} \end{aligned}$$

$$\text{We get } \arg\min_{\mathrm{w}} f(\mathbf{w}) = \sum_{i=1}^{n} \left(\mathbf{y}_i - \mathbf{x}_i\mathbf{w}^\top\right)^2 + \lambda \sum_{j=1}^{d} |\mathbf{w}_j| = \left\|\mathbf{y} - \mathbf{X}\mathbf{w}^\top\right\|_2^2 + \lambda\|\mathbf{w}\|_1$$

Substitute the prior distribution of w to $\mathcal{N}(0, \tau^2)$,we get Ridge regression form. Compare with two prior distribution, it is not difficult to find that Laplace is more steeper meaning that Lasso is more likely to prune the model, that is, penalize the parameter to zero .

2.3.3 Feature Selection Interpretation

As discussed in the Introduction, Lasso is famous for its model interpretation ability because its feature structure is sparse. In other word, not all the entry of the parameter w is non-zero and the estimator will select certain feature from original predictor X.

So whether Lasso is a better method depends on whether a sparse feature structure or feature selection is required. As for the reason for Lasso preferring such structure we will discuss it more sufficiently in the Section 2.4.

2.4 Comparison in Closed-form(Analytical) Solution

OLS(Ordinary Least Square) The OLS method minimizes the sum of squared residuals, and leads to a closed-form expression for the estimated value of the unknown parameter vector β.

First, we consider the optimization problem

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 \right\}$$

without any constrains. Simple calculation of derivative leads to the equation

$$-2X^T(y - X\beta) = 0$$

then we get the closed-form solution of OLS:

$$\beta^{OLS} = \left(X^T X\right)^{-1} X^T y \tag{6}$$

If $X^T X$ is singular, one can replace $(X^T X)^{-1}$ by Moore-Penrose(General) inverse $(X^T X)^{-}$.

Ridge Regression Ridge regression, also known as Tikhonov regularization, can be included in this minimization:

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \right\}$$

The derivative of the object gives the equation

$$-2X^T(y - X\beta) + 2N\lambda\beta = 0$$

then we get the closed-form solution of Ridge regression:

$$\beta^{\text{Ridge}} = \left(X^T X + N\lambda I\right)^{-1} X^T y \tag{7}$$

Lasso Regression Lasso regression is included in (3), which has an L_1 penalty. Therefore, derivative in Lasso regression is a little more complicated.

First, we have

$$\frac{2}{N}\left(X^T X\beta - X^T y\right) + \lambda \frac{\partial \|\beta\|_1}{\partial \beta} = 0 \tag{8}$$

One problem is that the derivative of the L_1 norm isn't well defined, hence we would introduce sub-gradient. Assuming that the covariates are orthonormal, so that $(x_i, x_j) = \delta_{ij}$, or equivalently $X^T X = I$. Using sub-gradient method, we need to consider the following cases:

1) If $\beta_j^{OLS} > \frac{N\lambda}{2}$, then we have $\beta_j = \beta_j^{OLS} - \frac{N\lambda}{2}\frac{\partial\|\beta\|_1}{\partial\beta_j}$. Since $\frac{\partial\|\beta\|_1}{\partial\beta_j} \leq 1$, we have $\beta_j > 0$, thus $\beta_j = \beta_j^{OLS} - \frac{N\lambda}{2}$.

2) If $\beta_j^{OLS} < -\frac{N\lambda}{2}$, then we have $\beta_j = \beta_j^{OLS} - \frac{N\lambda}{2}\frac{\partial\|\beta\|_1}{\partial\beta_j}$. Since $\frac{\partial\|\beta\|_1}{\partial\beta_j} \geq -1$, we have $\beta_j < 0$, thus $\beta_j = \beta_j^{OLS} + \frac{N\lambda}{2}$.

3) If $-\frac{N\lambda}{2} \leq \beta_j^{OLS} \leq \frac{N\lambda}{2}$, our plan is to show that $\beta_j = 0$. We can prove in contradiction. Assume that $\beta_j > 0$, then $\frac{\partial\|\beta\|_1}{\partial\beta_j} = 1$, thus $\beta_j = \beta_j^{OLS} - \frac{n\lambda}{2} \leq 0$, which leads to a contradiction. Follow the same argument, we can prove that the assumption $\beta_j < 0$ also leads to a contradiction. In other words, we have proved that $\beta_j = 0$.

In conclusion of three cases, we would get the closed form solution of Lasso regression problem:

$$\beta_j^{\text{Lasso}} = \text{sign}\left(\beta_j^{OLS}\right)\left(\left|\beta_j^{OLS}\right| - \frac{N\lambda}{2}\right)_+ \tag{9}$$

And Compare (9) and (7) we see the minor entry of β (means $|\beta_j^{OLS}| < \frac{N\lambda}{2}$) is compressed to zero in Lasso, which explains why the feature structure of Lasso is sparse.

Principal Component Regression [3]

Principal component regression (PCR) is based on principal component analysis (PCA). Instead of regressing the dependent variable on the explanatory variables directly, the principal components of the explanatory variables are used as regressors. We typically use only a subset of all the principal components for regression. Then we can perform OLS on these principal components. In this paper, instead of the number of nonzero entries, the size of the subset of principal components is considered as 'feature dimension' of PCR.

Remark Why not solve those regression problems directly by closed-form? Actually, when the dimension of X is large, it's time consuming, or even beyond the CPU's ability, to compute the inverse of $X^T X$! The intention to avoid computing the inverse gives motivation to exploit various algorithms, which is shown in the next section.

2.5 Practical Algorithm

In this section, we would discuss the algorithms for the Lasso introduced by Friedman et al. [2010]. Before illustrating the algorithm, we need to derive the coordinate-wise update first.

Consider the usual setup for linear regression(1) and we have N observation pairs (x_i, y_i). Without loss of generality, we assume that the x_{ij} are standardized: $\sum_{i=1}^N x_{ij} = 0$, $\frac{1}{N}\sum_{i=1}^N x_{ij}^2 = 1$, for $j = 1, ..., p$. The Lasso solves the following problem[4]

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[\frac{1}{2N}\sum_{i=1}^N \left(y_i - \beta_0 - x_i^T\beta\right)^2 + \lambda\sum_{j=1}^p |\beta_j|\right]$$

[3]It was acknowledged by Jolliffe [1982] that Kendall (1957) suggested it in his book on Multivariate Analysis, as did Hotelling (1957) in an artical in the same year, and a well known example was given by Jeffers (1967). This paragraph is extracted from Park [1981]

[4]we simply set $\alpha = 1$ in the elastic net to gain an l_1 penalty

Denote the objective function in the minimization by $R(\beta_0, \beta)$. We would like to compute the gradient(or generally the sub-gradient) at $\beta_j = \tilde{\beta}_j$. Following the work of closed-form solution, we consider three cases:

1) If $\beta_j > 0$, from case 1) of the closed-form solution, we have $\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \beta_0 - \sum_{l\neq j} x_{il}\beta_l) \geq \lambda$, thus

$$\begin{aligned}\frac{\partial R}{\partial \beta_j} &= -\frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - \beta_0 - \sum_{l=1}^{p} x_{il}\beta_l\right) + \lambda \\ &= -\frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - \beta_0 - \sum_{l\neq j} x_{il}\beta_l\right) + \beta_j + \lambda\end{aligned}$$

Simply set the partial derivative to zero and we have

$$\beta_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - \beta_0 - \sum_{l\neq j} x_{il}\beta_l\right) - \lambda$$

2) If $\beta_j < 0$, similarly we have

$$\beta_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - \beta_0 - \sum_{l\neq j} x_{il}\beta_l\right) + \lambda$$

3) If $\beta_j = 0$, then from case 3) in the work of the closed-form, we have $-\lambda \leq \frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \beta_0 - \sum_{l\neq j} x_{il}\beta_l) \leq \lambda$. In this special case we simply set $\beta_j = 0$.

In conclusion of three cases, the coordinate-wise update is denoted by a soft-threshold function

$$\tilde{\beta}_j \leftarrow S\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - \tilde{y}_i^{(j)}\right), \lambda\right), \tilde{\beta}_0 \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left(y_i - x_i^T\tilde{\beta}\right) \tag{10}$$

where

- $\tilde{y}_i^{(j)} = \tilde{\beta}_0 + \sum_{l\neq j} x_{il}\tilde{\beta}_l$
- $S(z, \gamma)$ is the soft-threshold operator with value

$$\text{sign}(z)(|z| - \gamma)_+ = \begin{cases} z - \gamma & \text{if } z > 0 \text{ and } \gamma < |z| \\ z + \gamma & \text{if } z < 0 \text{ and } \gamma < |z| \\ 0 & \text{if } \gamma \geq |z| \end{cases} \tag{11}$$

Therefore, the coordinate-wise update is similar to the closed-form solution (9).Next we will discuss how to utilize (7) to derive our practical algorithms.

Naive Updates From coordinate-wise update (10), we see that

$$\begin{aligned} y_i - \tilde{y}_i^{(j)} &= y_i - \widehat{y}_i + x_{ij}\tilde{\beta}_j \\ &= r_i + x_{ij}\tilde{\beta}_j \end{aligned} \tag{12}$$

where $\hat{y}_i = \tilde{\beta}_0 + \sum_{l=1}^{p} x_{il}\tilde{\beta}_l$ is the current fit of the model for observation i, and hence $r_i = y_i - \hat{y}_i$ is the current residual. Thus we derive our updating formula:

$$\frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - \tilde{y}_i^{(j)}\right) = \frac{1}{N}\sum_{i=1}^{N} x_{ij} r_i + \tilde{\beta}_j \tag{13}$$

Remark The first term on the right-hand side is the gradient of the loss with respect to β_j. Coordinate descent is computationally efficient, because many coefficients are zero, and remain zero after each update, hence nothing needs to be changed. On one hand, the sum to compute the gradient costs $O(N)$ operations. On the other hand, if a coefficient changes after the update, then r_i is changed in each of $O(N)$ operations and the step costs $O(2N)$. Therefore, a complete cycle through all p variables costs $O(pN)$ operations.

Covariance Updates However, further efficiencies can be achieved by transforming (13). We can rewrite the first term on the right(up to a factor $1/N$) as

$$\sum_{i=1}^{N} x_{ij} r_i = \langle x_j, y\rangle - \sum_{k:\left|\tilde{\beta}_k\right|>0} \langle x_j, x_k\rangle \tilde{\beta}_k \tag{14}$$

where $\langle x_j, y\rangle = \sum_{i=1}^{N} x_{ij} y_i$. Concluding (13) and (14), we gain another updating method which is called Covariance Updates for we use covariance calculations during our iterations:

$$\frac{1}{N}\sum_{i=1}^{N} x_{ij}\left(y_i - \tilde{y}_i^{(j)}\right) = \frac{1}{N}\langle x_j, y\rangle - \frac{1}{N}\sum_{k:\left|\tilde{\beta}_k\right|>0} \langle x_j, x_k\rangle \tilde{\beta}_k + \tilde{\beta}_j \tag{15}$$

Remark The higher computation efficiency can be explained by noting that owing to the sparsity of β, many terms $\langle x_j, x_k\rangle \tilde{\beta}_k$ on the right of (14) are zeros, speeding up the iteration. Precisely, we need to compute inner products of each feature with y initially, and each time a new feature x_k enters for the first time, we need to compute and store its inner products with all the rest of the features ($O(Np)$ operations). If one of the coefficients currently in the model changes, we can update each gradient (the left-hand side of (14)) in $O(p)$ operations. Hence with m non-zero terms in the model, a complete cycle costs $O(pm)$ operations, since no new variables will become non-zero under soft-threshold operator, and costs $O(Np)$ for each new variable entered. As a result, $O(N)$ calculations in each step of Naive Updates are not necessary.

Combining with (10) (13) and (15), the practical algorithm can be described in summary:

3 Experiment

3.1 Discussion of the Effect of Lasso

In this section, we are aimed at answering Problem 1, that is, what's the effect of Lasso and consider

Algorithm 1 Coordinate Descent Algorithm for Lasso

```
Input: response vector $y$, standardized predictor matrix $X$
Output: the best penalty factor $\hat{\lambda}$, the parameter pair $(\beta_0, \beta)$
Step 1: Select the best penalty factor $\hat{\lambda}$ (corresponds with the least CV error) by cross-validation.
  for all $\lambda$ decreasing from $\lambda_{max}$ to $\lambda_{min}$ in log scale do
    while convergence condition do
      Update $(\beta_0, \beta)$ using the current parameters $(\tilde{\beta}_0, \tilde{\beta})$ and the coordinate-wise update (10) (replace the first term in the soft-threshold operator by Naive Updates (13) or Covariance Updates (15))
    end while
  end for
Step 2: Compute the parameters $(\beta_0, \beta)$ using $\hat{\lambda}$ and the nested loop in Step 1.
  return $\hat{\lambda}$, $\beta_0$, $\beta$
```

3.1.1 Simulation Setup

For simulations , we test our codes on the example that the $n \times p$ predictor matrix X is generated from a multivariate normal distribution with correlation between x_i and x_j being $0.5^{|i-j|}$, true model coefficient β is set as a vector of length p whose non-zero elements are first p_1 entries and finally the response vector y is generated from $\mathcal{N}(X\beta, I)$.

3.1.2 Lasso vs Other Regression Method

Better at Handling Sparse Structure Model with Feature Selection [5] To change the sparsity of our structure, we change the ratio of nonzero entries of the parameter while fix n and p. In Figure 2, Lasso performs well when $p1/p$ is small, which means it is better at handling sparse structure model. On the other hand, the error of Lasso grows rapidly with p_1/p tends to 1, while PCR keeps stable.

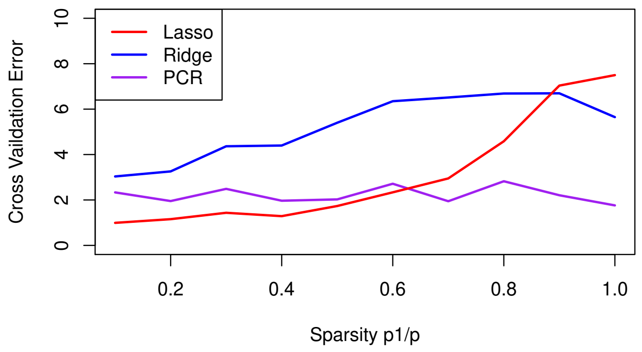

Figure 2: *Different rate of sparsity* $p1/p$ *,with* $n = 200$ *and* $p = 100$

[5]In this paper we understand **sparse structure** as a regression model with sparse real estimator β, that is, many entries of β are zero.

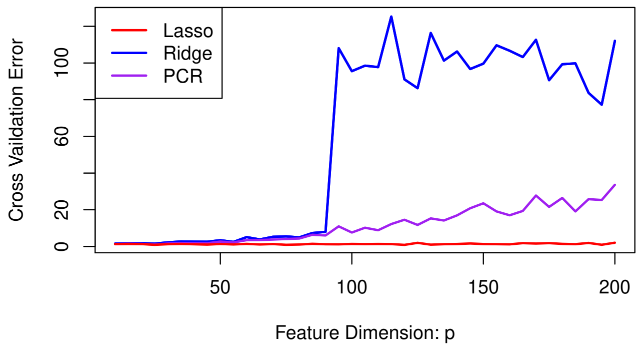

Figure 3: *Different feature dimension p of X, with sample number $n = 100$*

Able to Work when $p > n$ and Keep Stable when p is Close to n We keep $n = 100$ and change p (from 10 to 200), getting the 10 fold cross-validation error of three methods with fixed non-zero elements of $\beta = (1, 1, 4, 5, 1, 4, 0.1, 1.9, 1.8, 3)$.The simulation result is demonstrated on Figure 3 that the cross validation of Lasso stays low regardless of the change of p but Ridge and PCR methods behave bad when $p > n = 100$ and not stable even when p is close to n.

The reason for this result is that if $p > n$, which means the samples are not enough to solve the linear problem, the regression models may fail. However, if the size of the support set of β is limited, and smaller than the sample size, it would be possible to solve the problem by feature selection. In other words, l_1 penalty identifies the nonzero items of parameter, then we can fit the model in the case that the dimension of parameter is smaller than the sample size. That's the reason why Lasso keeps stable when $p > n$ with $\text{supp}(p) < n$.

Also Able to Deal With the Non-sparsity Structure with Enough Samples The true parameter of Figure 4 is $\beta = (\underbrace{4, ..., 4}_{p/2}, \underbrace{2, ..., 2}_{p/2})$, which is not sparse at all. However, when sample size n is large enough, Lasso performs as well as Ridge, while PCR performs even better. The result leads to the conclusion that Lasso is also strong in non-sparse condition when enough samples can be given.

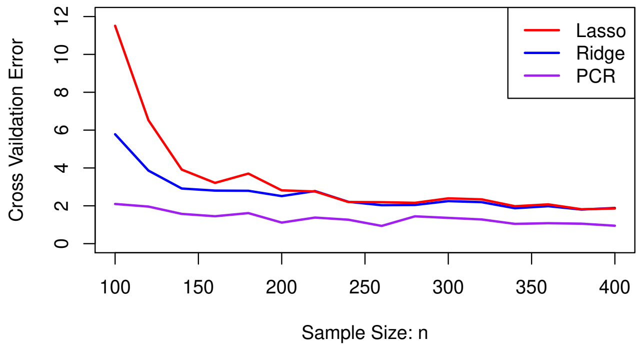

Figure 4: Different sample size n, with feature dimension $p = 50$

3.2 Discussion of the Efficiency of Solving Lasso Problem

To answer Problem 2, we analyze why *glmnet* package enjoys fast optimization speed by conducting some comparing simulations on the regression problem 2. It is worthy of noting that all these experiments are not dependent of any existed R package for we want to control the variables and exactly evaluate the method used in the *glmnet* package.

3.2.1 Simulation Setup

For next two simulations, we test our codes on the example that the 150$\times$ 90 predictor matrix X ($n = 150, p = 90$) is generated from a multivariate normal distribution with correlation between x_i and x_j being $0.5^{|i-j|}$, true model coefficient β is set as a vector of length 90 whose non-zero elements are first 9 entries (1,1,4,5,1,4,1,1,4) and finally the response vector y is generated from $\mathcal{N}(X\beta, I)$ and for simplicity we set the penalty parameter $\lambda = 0.5$ instead of using cross-validation. We stop the loop until every the update of all the coordinate is less than 1×10^{-6}.

3.2.2 Naive Update vs. Corvariance Update

First we compare two updating method mentioned in the previous section, Naive Updates (13) and Covariance Updates(15) and show the result in the Table 1.

Table 1 depicts that as the previous section analyze, the Covariance Updates are faster to Naive Updates obviously(almost 10 times) but get the same estimate. And for comparison, we use *glmnet* Package to do the same simulations and find that *glmnet* is faster in the loss descending speed per iteration as excepted but the gap between two method is not big, considering that all three methods correctly select the non-zero element of true β .

Table 1: Result of Naive Update and Covariance Update ,$\lambda = 0.5$

Method	Loss	Running Time (s)	Cardinality of β
Naive Update	0.0128	7.171	9
Covariance Update	0.0128	0.263	9
glmnet Package	0.0127	0.043	9

3.2.3 Soft-Thresholding Method vs Other Approximation Method

In this part, we discuss the efficiency of the soft-thresholding method (11) by comparing it with other two methods that deal with the absolute value function, including **Huber Loss** and **Proximal Gradient Descent**

Huber Loss Approximation Huber loss function is a C_1 continuous function defined on $\mathbb{R}$, which is a piece-wise combination of quadratic and absolute value function. Its math definition is below:

$$g_\delta(x) \begin{cases} \frac{x^2}{2\delta} & \text{if } x \in [-\delta, \delta] \\ |x| - \frac{\delta}{2} & \text{otherwise} \end{cases}$$

Note that huber loss function converge to the absolute function if δ tends to 0, and has deviate on the origin so we can use huber loss to approximate $|x|$ used in Lasso. In this simulation, we set δ tends to 0 with the increasing iteration times and we also reduce the step size at the same time so that it can converge.

Proximal Gradient Descent [6] We consider the compound optimization problem:

$$\min \psi(x) = f(x) + h(x)$$

where f is smooth and h is convex but not necessary to be smooth. To deal with the singularity of h, we define proximal mapping:

$$\text{prox}_h(x) = \arg \min_{u \in \mathbf{dom} f} \left\{ h(u) + \frac{1}{2}\|u - x\|^2 \right\}$$

The idea of proximal gradient method is very simple: notice that ψ has two components, we apply gradient descent to f, and use proximal operator on h, then we derive the update fomula:

$$x^{k+1} = \text{prox}_{t_k h}\left(x^k - t_k \nabla f(x^k)\right) \tag{16}$$

where t_k is the step of each iteration, which can be a constant or selected by line search method. In Lasso problem, $f(x) = \frac{1}{N}\|y - X\beta\|_2^2$, and $h(x) = \lambda\|\beta\|_1$, then simple argument shows that

$$\nabla f(x) = \frac{2}{N} X^T (X\beta - y), \quad \text{prox}_{t_k h}(x) = \text{sign}(x)\left(|x| - t_k u\right)_+ \tag{17}$$

Since f is gradient L-Lipschitz continuous, combining (16) (17) with the convergence rule of line search method, we derive the algorithm of proximal gradient descent:

[6]This paragraph is extracted from Mosci et al. [2010]

Algorithm 2 Proximal Gradient Method for Lasso

```
Input: Initial parameter $(\beta_0, \beta)$, response vector $y$, standardized predictor matrix $X$
Output: the parameter pair $(\hat{\beta}_0, \hat{\beta})$ for Lasso
  $L = \frac{2}{N}\|X^T X\| = \frac{2}{N}\lambda_{\max}$, where$\lambda_{max}$ is the max single value of $X^T X$
  while convergence condition do
    $\eta^k = \beta^k - \frac{2}{NL}X^T(X\beta^k - y)$
    $\beta^{k+1} = \text{sign}(\eta^k)(|\eta^k| - \frac{\lambda}{L})_+$
    $\beta_0^{k+1} = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i^T\beta^{k+1})$
  end while
  return $\beta_0$, $\beta$
```

As demonstrated in the Table 2, we arrive at our conclusion that soft-thresfolding method is faster than other two methods. It also reflects from simulation that *glmnet* enjoys high computation efficiency for the usage of coordinate descent and soft-thresfolding method.

Table 2: Result of Soft-Thresholding Method and Other Approximation Methods, $\lambda = 0.5$

Method	Loss	Running Time (ms)	Cardinality of β	Iterations
Soft-Thresholding(Cov Update)	0.0128	0.263	9	25×90
Proximal Gradient Descent	0.0128	4.429	9	15289
Huber Loss	0.0164	8.642	90	718×90

3.3 Experiment on the Real Data Set

In this section, we do experiment on two real data set[7], and compare the run times of *glmnet* to *pcr* (principal component regression)[8]. These use Lasso penalty ($\alpha = 1$) and ridge penalty ($\alpha = 0$) in regression settings. Table 4 and 5 below show the results of tests on data set *mushrooms* [9] and *rcv1* [10] .

3.3.1 Preprocess the Source Data

Table 3: Dimension of mushrooms and rcv1

Data Set	row	column
mushrooms	8124	112
rcv1	8000	300

The source data *mushrooms* and *rcv1* is shown in Table 3. However, *rcv1* has many rows and columns that are filled with zero. Therefore, we need to remove those rows and columns in

[7]The two data sets are also used by Perekrestenko et al. [2017]

[8]The functions *pcr* and *pcr.cv* are in R package *plsdof*

[9]`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/mushrooms`

[10]`rcv1_train.binary`

the data set at first, then we get a 2661×182 matrix A and a column vector y of dimension 2661.
It's obvious that both data set satisfy the condition that the sample size n is much larger than feature dimension p. Following the conclusion of section 3.1.2, the errors of these three methods should be almost equal.

3.3.2 Results of the Tests on Two Data Sets

Table 4: Result of cv.glmnet and pcr.cv on mushrooms data set

Method	cv.error	Running Time (s)	Features Selected
glmnet-lasso	0.001349421	1.25	63
glmnet-ridge	0.002451355	1.25	111
pcr	0.001273011	4.61	61

Table 5: Result of cv.glmnet and pcr.cv on rcv1 data set

Method	cv.error	Running Time (s)	Features Selected
glmnet-lasso	0.8158304	1.01	141
glmnet-ridge	0.8566097	1.03	182
pcr	0.8614481	4.2	182

As is shown in those two tables, it is obvious that *glmnet* has higher computing efficiency than *pcr*, while their cv.errors, in other words the precision, are nearly the same.
On one hand, the errors are small enough, which means both of them are practical. On the other hand, since the functions are designed based on different algorithm, the results would have different features, more precisely, the index such as sparsity and weight may differ. Therefore, one can choose either of them in practice to gain expected results.

4 Conclusions

In this paper, we analyze Lasso theoretically and conduct relevant simulations and experiments in order to answer Problem 1 and 2.

In the theory section, we interpret the Lasso form from different views and derive the closed-form solution , from which we know that soft thresholding operator coerces minor parameters into zero so the solution of Lasso is usually sparse. We also analyze the practical algorithm used in *glmnet* Package, discussing the reasons for its outstanding performance.

For the simulations part, to discuss the effect of Lasso by compared with Ridge and PCR, we use cross validation error to measure the precision of different models, and change sample size, feature dimension and ratio of sparsity to verify the properties of Lasso.

And we focus on the methodology of Lasso problems. In comparison, we run simulations on generated data. Firstly, we compare Naive Updates and Covariance Updates with self-implemented codes, as well as *glmnet* package. Furthermore, the second simulation is CD with soft thresholding, where we utilize Covariance Updates, versus other methods. And we also test *glmnet* package and *plsdof* package on real data sets as comparison.

Then we can come to the answer of the two problems raised in the Introduction section. For Problem 1, sparsity structure of solution is a significant advantage of l_1 penalty, which help us find out the variables that have great influence on the results, and exclude inference of other variables. For problem 2, we find that CD with soft-thresholding(Corvariance Updates) method enjoys higher computing efficiency comparing to CD with Huber loss and Proximal Gradient Descent, explaining the state-of-art performance of *glmnet* Package.

Appendix A Introduction to R Package *glmnet*

To satisfy the requirement and avoid interrupting the stream of our main content, we introduce some important functions in this R package in the appendix part.

cv.glmnet The most important function in practice.

- Parameter:
  - x x matrix as in *glmnet*
  - y response y as in *glmnet*
  - foldid an optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfold can be missing.
- Value:
  - lambda.min value of lambda that gives minimum cvm.
  - lambda the values of lambda used in the fits.
  - cvm the mean cross-validation error, a vector of length (λ)
  - glmnet.fit a fitted glmnet object for the full data.

glmnet The function to solve the lasso problem.

- Parameter:
  - x input matrix, of dimension nobs x nvars; each row is an observation vector. Can be in sparse matrix format (inherit from class "sparseMatrix" as in package Matrix; not yet available for family="cox")
  - y response variable.
  - family Response type. Either a character string representing one of the built-in families("gaussian", "poisson", "binomial", "multinomial", "cox", "mgaussian"), or else a glm() family object.
- Value:
  - lambda The actual sequence of lambda values used.

Remark: How to choose an appropriate λ is an important issue:

- If we use cross validation, then we can just compare the MSE or PMSE of different alternative λ.
- If we don't use cross validation, we can select λ by its proportional deviance explained[11]($\%dev$). In practice, we choose the λ with highest $\%dev$ as our best λ.

References

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

Ian T. Jolliffe. A note on the use of principal components in regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(3):300–303, 1982. ISSN 00359254, 14679876. URL `http://www.jstor.org/stable/2348005`.

Sofia Mosci, Lorenzo Rosasco, Matteo Santoro, Alessandro Verri, and Silvia Villa. Solving structured sparsity regularization with proximal methods. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 418–433, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15883-4.

Sung H. Park. Collinearity and optimal restrictions on regression parameters for estimating responses. *Technometrics*, 23(3):289–295, 1981. ISSN 00401706. URL `http://www.jstor.org/stable/1267793`.

Dmytro Perekrestenko, Volkan Cevher, and Martin Jaggi. Faster coordinate descent via adaptive importance sampling. In *Artificial Intelligence and Statistics*, pages 869–877. PMLR, 2017.

Jon M Sutter, John H Kalivas, and Patrick M Lang. Which principal components to utilize for principal component regression. *Journal of chemometrics*, 6(4):217–225, 1992.

Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[11]Like R squares, which is defined as $\mathrm{R}^2 = 1 - \frac{\text{Unexplained Variation}}{\text{Total Variation}}$