On LASSO for autocorrelated errors

Shweta Shaw

Abstract. With the advent of modern technologies, most of the data we encounter contains informations on thousands of variables. In the context of regression, this large number of potential covariates often causes difficulties in finding an optimal yet parsimonious model which can be interpreted easily. However, in the recent years LASSO has gained popularity as one of the efficient tools for high dimensional regression modeling. One major reason is that LASSO performs prediction and variable selection in a very systamatic way. Also it performs better than ordinary least square in terms of MSE for suitably choosen shrinkage operator. But so far what we have studied about LASSO the main assumption in fitting a model, is errors are independent. Here we want to study how LASSO will perform under correlated errors.

Contents

CHAPTER 1

Curse of dimensionality

1.1. Introduction

The linear regression model is a commonly used statistical tool to study the relationships between response and explanatory variables. Clearly speaking, linear regression belongs to the class of parametric regression. One of the potential problems with parametric regression is the increase in number of covariates or explanatory variables. Even if all the assumptions of linear regression hold good least square method cannot be applied if $p > n$. Further even if $p < n$, but large enough, linear regression model can be defined but extremely hard to interpret. Also with the increase in number of covariates, there is a potential threat of multicollinearity. The problems that we face due to increase in dimension is nothing but curse of dimensionality.

1.2. Curse of dimensionality in regression

Due to increase in number of covariates or in other words increase in dimension of the data we face certain problems. The reasons for these problems are the following :

- ***Prediction Accuracy :*** With increased number of covariates, we have a large number of parameters in the model. This will lead to low bias but high variance, so that the overall prediction accuracy (in terms o MSE) will be degraded.
- ***Interpretation :*** Including a large number of covariates in the model makes the model more flexible but hard to interpret. Infact most of the covariates in such cases are found to have insignificant effects on the response.
- ***Multicollinearity :*** When we include many covarites in the model, there is a high chance that most of the covariates are found to be dependent among themselves. This causes the parameters in the model unidentifiable and other problems related to multicollinearity such that standard error of the estimates of correlated parameters will be very high so that the estimates will not be reliable.

1.3. Some possible solutions

To solve the above mentioned problems we need to reduce the dimension of the data. The approaches to solve the problem can be classified in three categories :

(i) Derived inputs

(ii) Variable selection

(iii) Shrinkage

- ***Derived Inputs :*** In this approach we can apply principal component technique to the covariates and choose k (<<p) linear combinations as our new covariates to perform the regression. This method is called Principal Component Regression, where k linear combination of covariates are the new covariates that captures the inherent variability of the whole set of covariates. But the general problem with this approach is to interpret the model. As we are taking the linear combination of covariates the significance of individual covariate will be lost.
- ***Variable Selection :*** This approach involves identifying a subset of the p predictors that we believe to be highly related o response. Then we fit a model using least squares on the reduced set of variables.
- ***Shrinkage :*** In this approach we fit a model using all covariets . However, the estimated coefficients are shrunken towards zero relative to the least squares estimates. This shrinkage (also known as regularization) has the effect of reducing variance. Depending on what type of shrinkage is performed, some of the coefficients be shrunken to be exactly zero. One of such shrinkage method is known as LASSO or Least Absolute Selection and Shrinkage Operator.

1.4. Objective

Our aim is to use lasso in regression model with dependent errors. The main application of this idea is time-lagged regression, where we predict an outcome at time t from the predictors at previous q time points. Note that, regression model with autoregressive errors are considered here, so the prediction at time t will not only depend on previous q time points but also on the corresponding covariates at all these time points. Here first autoregressive process of order 1 is considered and how to incorporate lasso in that setup, is described. Then we move on to a general setup where the process is autoregressive of order q. The solution to this problem is descrided and also its drawbacks. Finally to overcome these drawbacks of the previous problem Ordered Lasso is used to estimate autoregression parameters of autoregressive process of order q.

1.5. Outline of the work

In this project chapter 2 contains brief destription of LASSO and in what conditions lasso performs better than ordinary least square. Section (2.1) and (2.2) contains both the shrinkage methods, ridge and lasso. In figure (2.3.1),(2.3.2), (2.3.3), (2.3.4) we show that lasso performs better than OLS when some coefficients are very small for particular choices of shrinkage parameter. Chapter 3 is all about lasso for correlated error. In (3.1) we discuss the regression in general dependent structure of error. In (3.2) we assume that error following autoregressive process of order 1 and discuss the way to solve the problem. In section (3.3) we discuss how we will do lasso if errors are autoregressive process of order q. The problems that we face in section (3.3) has been discussed in (3.4) solved in (3.5) and (3.6) where, we introduce the concept of ordered lasso and tackle the lasso problem for time-lagged observation using ordered lasso. Chapter 4 is all about analysing a real life data with time lagged observation and using methods that discussed in chapter 3 we get lasso estimates of parameters and prediction accuracy in each case.

CHAPTER 2

LASSO

2.1. Introduction

In regression model if we have large number of covariates and we want to reduce the dimension of the data then we have a lot of methods in hand ,that has already been diccussed in (1.3). Here we want to discuss about shrinkage class of methods. Shrinkage is nothing but structural risk minimization problem. Structural risk minimization includes a penalty function that controls the bias vs variance tradeoff. Structural risk minimization tries to pervent overfitting by incorporating a regularization penalty criterion. In case of linear regression we have function of the form $f(x) = X\beta$. One can incorporate a regularization penalty $C(f) = \sum \beta_j^2$ or $C(f) = \sum |\beta_j|$. Then we can try to minimize

$$J(f) = R(f) + \lambda C(f)$$

where $R(f) = ||y - f(x)||_2^2$. For $C(f) = \sum \beta_j^2$ we will have ridge regression and for $C(f) = \sum |\beta_j|$ we will have lasso solution. Here $\lambda \geq 0$ is the tuning parameter that controls the robustness of penalty. For $\lambda = 0$ we have least square estimate but as we increase λ more coefficients will be shrunken towards zero. This tuning parameter actually controls bais vs variance tradeoff. For a small value of λ we will have a more flexible model which has low bais but high variance and for a large value of λ we will have a less flexible model which has low variance but high bias. So, changing the value of λ we can control model flexibility.

2.2. Penalized least squares

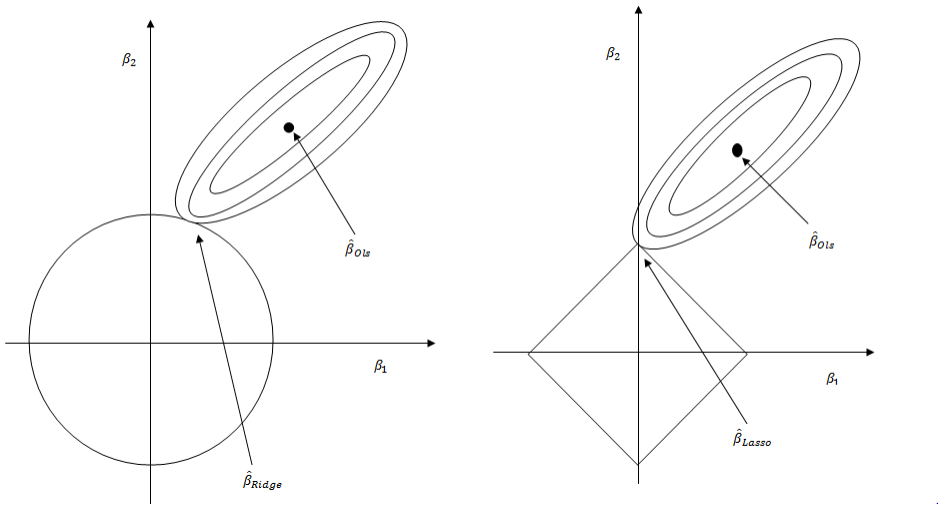
The lasso estimate is defined as

$$\begin{aligned}\hat{\beta}^{lasso} &= \arg\min_{\beta \in \mathbb{R}^p} \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \\ &= \arg\min_{\beta \in \mathbb{R}^p} \left\{ \underbrace{||y - X\beta||_2^2}_{\text{Loss}} + \lambda \underbrace{||\beta||_1}_{\text{Penalty}} \right\}.\end{aligned}$$

Thus the only difference between lasso and ridge regression is that in ridge model we use a squared ℓ_2 penalty $||\beta||_2^2$ and in lasso we use an ℓ_1 penalty term $||\beta||_1$. The change in this penalty term results the solutions of both to behave very much different. In ridge none of the coefficients will exactly be zero due to the nature if its penalized crierion but in lasso the insignificant variable will exactly reduced to zero. Here λ is a tuning parameter which controls the strength of penalty. For $\lambda = 0$, $\hat{\beta}^{lasso}$ is the OLS estimator $\hat{\beta}$ and for $\lambda = \infty$, $\hat{\beta}^{lasso} = 0$. For λ in between, we are balancing the two ideas: fitting a linear model of y on X and shrinking the coefficients towards zero. Moreover the nature of the shrinkage is

interesting. The ℓ_1 penalty causes some of the coefficients to be **shrunken to zero exactly**. This makes lasso important from the perspective of **variable selection**. With the increase in λ, more and more coefficients are set to zero, i.e. more and more covariates are removed from the model. Also among the nonzero coefficients, increase in λ causes more shrinkage.

FIGURE 2.2.1. Comparison between Ridge and Lasso constraints

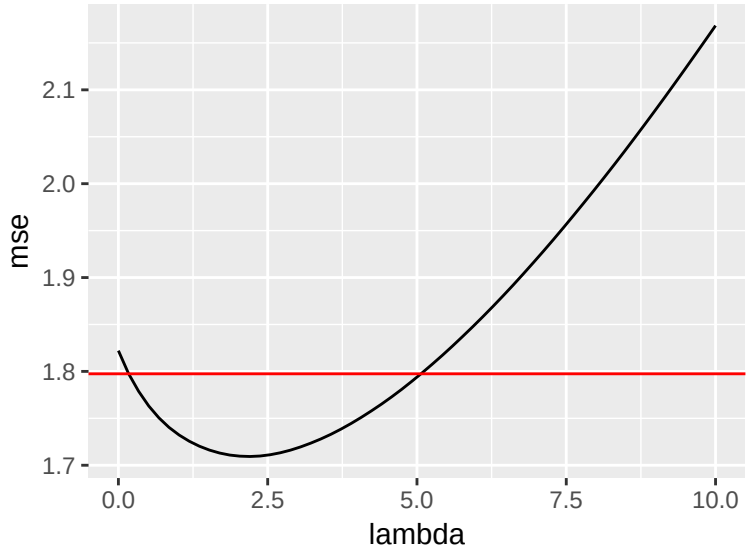


The Lasso constrained zone has "corner points" so it makes some coefficient exactly zero which is not possible in case of Ridge constrained zone

2.3. Optimal shrinkage

In terms of prediction accuracy LASSO can outperform ordinary least square if the shrinkage parameter is selected appropriately in fig. (2.3.1).

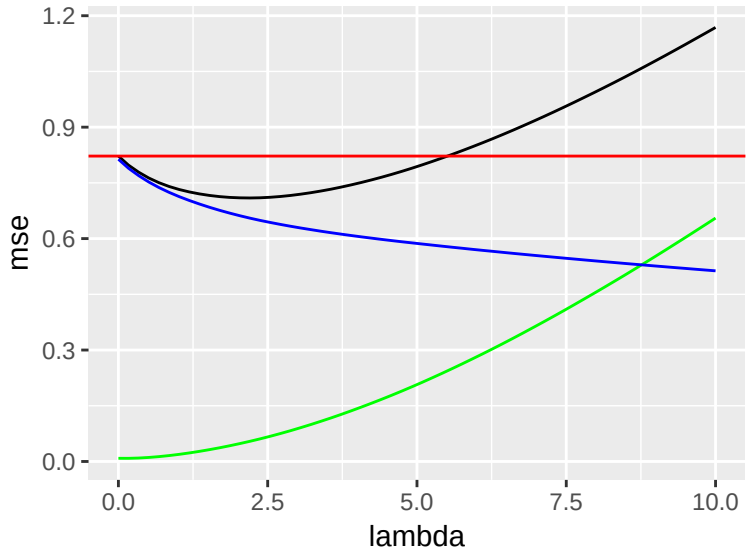
FIGURE 2.3.1. LASSO can perform better than OLS



MSE curve of OLS and LASSO
'Red' horizontal line is the MSE curve for OLS and 'black' U-shapped curve is MSE for LASSO

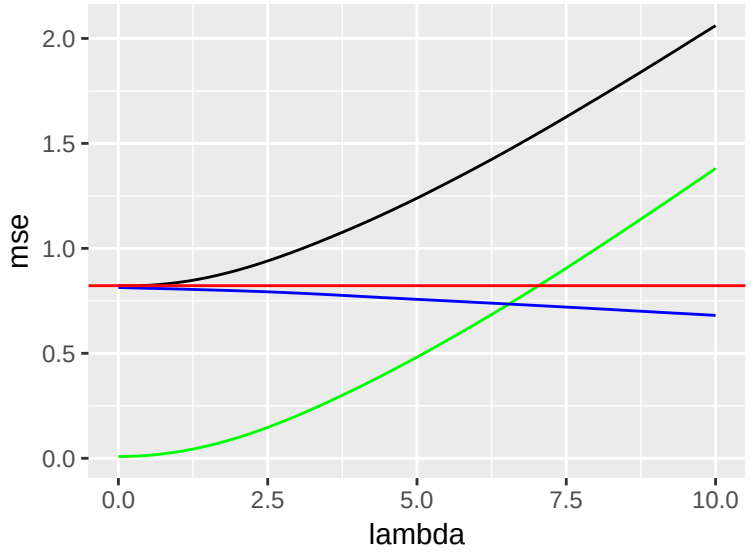
Bias and variance of the lasso: We can't write down explicit formulas for the bias and variance of the lasso estimate. But we know the general trend. The bias increases as λ (amount of shrinkage) increases. The variance decreases as λ (amount of shrinkage) increase. Here we in the simulated data we can see the nature of bias and varianace with increase in value of λ. First, among all coefficients a subset of coefficient are taken to be small in fig (2.3.2), then all coefficients significantly large in fig. (2.3.3) and lastly a subset of coefficient to be exactly zero in fig. (2.3.4). In all the case, it is seen that lasso performs better than ols for a certain range of λ and as we increase λ bias increases but variance decreases.

FIGURE 2.3.2. Bias and Variance in LASSO



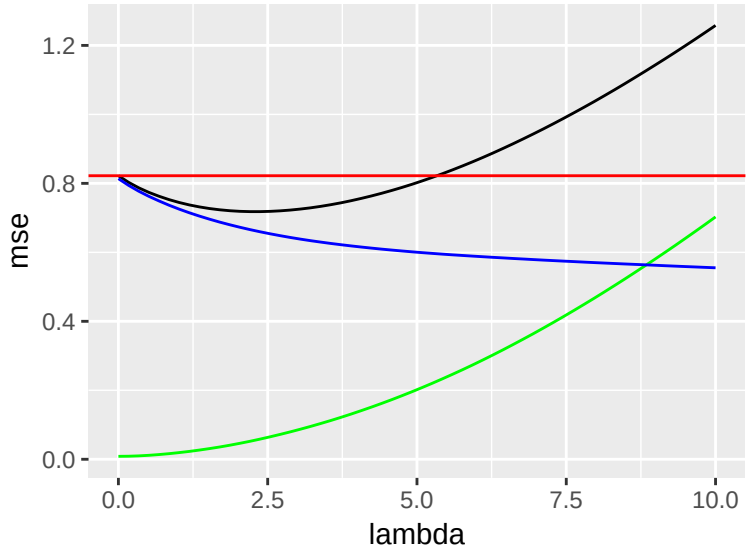
MSE, bias and variance curve for a subset of small coefficients 'Red' horizontal line is MSE curve for OLS, 'black' U-shapped curve is MSE curve, 'green' increasing curve is bias curve and 'blue' decreasing curve is variance curve for LASSO

FIGURE 2.3.3. LASSO when coefficients are significant



MSE, bias and variance curve when all coefficients are significant 'Red' horizontal line is MSE curve for OLS, 'black' U-shapped curve is MSE curve, 'green' increasing curve is bias curve and 'blue' decreasing curve is variance curve for LASSO

FIGURE 2.3.4. LASSO when some coefficients are zero



MSE, bias and variance curve when a subset of coefficient is exactly zero 'Red' horizontal line is MSE curve for OLS, 'black' U-shapped curve is MSE curve, 'green' increasing curve is bias curve and 'blue' decreasing curve is variance curve for LASSO

2.4. Co-ordinate descent Algorithm

Now we describe a method of computing the LASSO solution known as **co-ordinate descent**. For this we consider an alternative description of the lasso problem as

$$\underset{\beta}{\text{minimize}}\left\{\frac{1}{2n}||y - X\beta||_2^2 + \lambda||\beta||_1\right\}$$

We begin with a single predictor and then generalize the method for multiple predictors. We assume that the predictors are **standardized**.

Single Predictor: Soft Thresholding: Suppose we have a single covariate and the data is obtained as $\{(z_i, y_i)\}_{i=1}^n$. Then our problem reduces to

$$\underset{\beta}{\text{minimize}}\left\{\frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta z_i)^2 + \lambda|\beta|\right\}$$

We could have approached to minimize it by taking the gradient (i.e. derivative) w.r.t. β and set it to zero. However $|\beta|$ does not have a derivative at $\beta = 0$.

We find from the KKT conditions that any optimal solution should satisfy

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\beta}z_i)z_i + \lambda sign(\hat{\beta}) = 0$$

$$\Rightarrow -\frac{1}{n}\sum_{i=1}^{n} y_i z_i - \hat{\beta} + \lambda sign(\hat{\beta}) = 0$$

Further we note that $\hat{\beta}^{OLS} = \frac{1}{n}\sum y_i z_i$. These two facts imply

- If $sign(\hat{\beta}) > 0$, we have $\hat{\beta} = \frac{1}{n}\sum y_i z_i - \lambda = \hat{\beta}^{OLS} - \lambda$
- If $sign(\hat{\beta}) < 0$, we have $\hat{\beta} = \frac{1}{n}\sum y_i z_i + \lambda = \hat{\beta}^{OLS} + \lambda$
- If $sign(\hat{\beta}) = 0$, we have $\hat{\beta} = 0$

Thus we get :

- $\hat{\beta}(\lambda) = \hat{\beta}^{OLS} - \lambda$ if $\hat{\beta}^{OLS} > 0$ and $\lambda < \hat{\beta}^{OLS}$
- $\hat{\beta}(\lambda) = \hat{\beta}^{OLS} + \lambda$ if $\hat{\beta}^{OLS} < 0$ and $\lambda > -\hat{\beta}^{OLS}$
- $\hat{\beta}(\lambda) = 0$ if $\hat{\beta}^{OLS} = 0$ or $\lambda \geq |\hat{\beta}^{OLS}|$

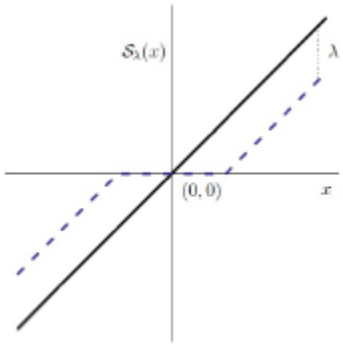
Now the case $\hat{\beta}^{OLS} = 0$ is included in the case $\lambda \geq |\hat{\beta}^{OLS}|$ (because $\lambda \geq 0$). Hence combining we get

$$\hat{\beta}(\lambda) = \begin{cases} \frac{1}{n}\langle z, y\rangle - \lambda & \text{if } \frac{1}{n}\langle z.y\rangle > \lambda \\ 0 & \text{if } \frac{1}{n}|\langle z.y\rangle| \leq \lambda \\ \frac{1}{n}\langle z, y\rangle + \lambda & \text{if } \frac{1}{n}\langle z.y\rangle < -\lambda \end{cases}$$

The soft thresholding operator is defined by

$$S_\lambda(x) = sign(x)(|x| - \lambda)_+$$

FIGURE 2.4.1. Soft Thresholding Operator



Soft Thresholding Operator pooled $y = x$ curve towards x axis by λ amount

This operator simply translates x towards 0 by the amount λ and sets it to zero if $|x| < \lambda$. Using the soft thresholding operator we can write the estimate of β as

$$\hat{\beta}^{Lasso} = \hat{\beta}(\lambda) = S_\lambda(\frac{1}{n}\langle z, y\rangle)$$

That is, $\hat{\beta}^{Lasso}$ is the soft thresholded version of the ordinary least square estimate $\hat{\beta}^{OLS} = \frac{1}{n}\langle z, y\rangle$.

Multiple Predictors: Following the same intuitive approach as in the case of one covariate, we now extend it for the general lasso problem. We repeatedly cycle through the predictors in some fixed (but arbitrary) order (say, $j = 1, 2, ..., p$) where at the j^{th} step we update the coefficient β_j by minimizing the objective function w.r.t. this coordinate while holding fixed other coefficients $\{\hat{\beta}_k, k \neq j\}$ at their current values. Rewrite the objective function as

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \sum_{k\neq j} x_{ik}\beta_k - x_{ij}\beta_j)^2 + \lambda\sum_{k\neq j}|\beta_k| + \lambda|\beta_j|$$

Also denote the partial residual as

$$r_i^{(j)} = y_i - \sum_{k\neq j} x_{ik}\hat{\beta}_k$$

From the objective function we can express the j^{th} coefficient in terms of the residual as

$$\hat{\beta}_j = S_\lambda(\frac{1}{n}\langle x_j, r^{(j)}\rangle)$$

Equivalently, the update can be written as

$$\hat{\beta}_j \leftarrow S_\lambda(\hat{\beta}_j + \frac{1}{n}\langle x_j, r\rangle)$$

where $r_i = y_i - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j$ are the full residuals.

The overall algorithm operates by applying this soft thresholding update repeatedly in a cyclical manner, updating each coordinate of $\hat{\beta}$ (and hence the residuals) along the way.

Validity of this algorithm : The lasso criterion is a strictly convex function of β and so has no local minima. The algorithm just described corresponds to the method of cyclical coordinate descent, which minimizes this convex objective along each coordinate at a time. Under relatively mild conditions (which apply here), such coordinate-wise minimization schemes applied to a convex function converge to a global optimum. Hence, the lasso criterion can be minimized using this algorithm and the minimized value of β will be the local minimum.

Further Details: In practice, we do not find the lasso solution not just for a single fixed value of λ. Rather we need to find the entire path of solutions over a range of λ. A reasonable method for doing so is to begin with a value of just large enough so that the only optimal solution is the all-zeroes vector. We can show this value is $\lambda_{max} = \max_j |\frac{1}{n}\langle x_j, y\rangle|$. Then we decrease by a small amount and run coordinate descent until convergence. Decreasing again and using the previous solution as a "warm start," we then run coordinate descent until convergence. In this way we can efficiently compute the solutions over a grid of values. This method is referred to as **pathwise coordinate descent.**

CHAPTER 3

LASSO under correlated errors

3.1. Introduction

We want to use least absolute shrinkage and selection operator (lasso) in regression model with dependent errors. Consider the linear regression model $y \epsilon \mathbb{R}^n$ on $X^{n \times p}$with correlated errors that is

$$y = X\beta + \epsilon$$

where ϵ has mean 0 and dispersion matrix Σ. Σ is of the form

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & . & . & . & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & . & . & . & \sigma_{2n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \sigma_{n1} & \sigma_{n2} & . & . & . & \sigma_{nn} \end{pmatrix}$$

Note that, here the number of parameters is very large. There are p unknown coefficients in β and n^2 unknown parameters in Σ. So, it is hard to estimate all the parameters in the model. Also note that, we have n number of observations and $n^2 + p$ number of parameters. Hence, we cannot have unique solutions of these parameters. So, we assume some structure to the error. First we assume that errors are following autoregressive process of order 1, then assume autoregressive process of order q to generalize the problem to some extent.

3.2. The beginning : Autoregressive model of order 1

Here we assume that our error are following autoregressive process of order 1. Our regression model is

$$y = X\beta + \epsilon,$$

where y is the response variable, X is the matrix of covariates, β is the coefficient vector that we want to estimate and ϵ is the error. Clearly these errors are following autoregression process of order 1. Hence we can write,

$$\epsilon_t = \rho\epsilon_{t-1} + z_t, t = 2, 3, ..., n \tag{3.2.1}$$

where z_t is gaussian white noise with mean 0 and variance σ^2.

Therefore, $Disp(\epsilon) = \sigma^2\Sigma$, where

$$\Sigma = \begin{pmatrix} 1 & \rho & \rho^2 & . & . & . & \rho^{n-1} \\ \rho & 1 & \rho & . & . & . & \rho^{n-2} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & . & . & . & 1 \end{pmatrix}$$

where $\rho$is unknown.

In order to shrink unnecessary coefficients to zero and get an estimate of parameters, we adopt the lasso criterion and minimize it with respect to the parameters

$$L_n(\beta) = (y - X\beta)^{'}\Sigma^{-1}(y - X\beta) + \lambda \sum_{i=1}^{p} |\beta_i| \tag{3.2.2}$$

where λ is the shrinkage parameter and p is the number of parameters in the regression model.

Note that, here we only shrink the regression parameters not the autoregressive coefficient. Suppose, we also shrink the autoregressive parameter ρ and its comes out to be 0 after applying lasso. Then our model will be of no use as we have already cosidered that errors are autoregressive of order 1.

Now, Σ is a p.d. matrix so we can write $\Sigma^{-1} = P^{'}P$, where

$$P = \begin{pmatrix} \sqrt{1-\rho^2} & 0 & 0 & . & . & . & 0 \\ -\rho & 1 & 0 & . & . & . & 0 \\ 0 & -\rho & 1 & 0 & .. & . & 0 \\ . & . & . & . & . & . & .. \\ . & . & . & . & . & . & : \\ 0 & . & . & . & 0 & -\rho & 1 \end{pmatrix},$$

by Cochrane–Orcutt method[**1**], suggested by Cochrane and Orcutt in 1949 in their paper "Application of Least Squares Regression to Relationships Containing Auto-Correlated Error Terms".

Now consider the transformed model

$$Py = PX\beta + P\epsilon$$

$$\Rightarrow \widetilde{y} = \widetilde{X}\beta + \widetilde{\epsilon}, \tag{3.2.3}$$

where $\widetilde{y} = Py, \widetilde{X} = PX$ and $\widetilde{\epsilon} = P\epsilon$.

Therefore, the lasso criterion reduces to

$$L_n(\beta) = (\widetilde{y} - \widetilde{X}\beta)^{'}(\widetilde{y} - \widetilde{X}\beta) + \lambda \sum_{i=1}^{p} |\beta_i| \tag{3.2.4}$$

To get estimates of β_i's and ρ we adopt a iterative procedure. We start with initial values of $\beta = \beta_0$and $\rho = \rho_0$, say. Then we minimize (3.2.4) applying lasso and using initial value of ρ to get an estimate of $\beta = \beta_1$, say and applying least square method and using initial value of β in equation (3.2.1) we get an estimate of $\rho = \rho_1$,say. We repeat this process again if $|\beta_0 - \beta_1| > 0.001$, using β_1 and $\rho_1$as the initial values. Finally we take $\hat{\beta} = \beta_{m+1}$ and $\overset{\wedge}{\rho} = \rho_{m+1}$ if $|\beta_m - \beta_{m+1}| < 0.001$.

3.3. Generalization: Autoregressive model of order q

Consider, the linear regression model

$$y = X\beta + \epsilon,$$

where $X^{n\times p}$is the covariate matrix and $\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ . \\ : \\ \beta_p \end{pmatrix} \in \mathbb{R}^p$ is the associated regression coefficient. In addition, ϵ_t follows autoregressive process of order q that is,

$$\epsilon_t = a_1\epsilon_{t-1} + a_2\epsilon_{t-2} + ... + a_q\epsilon_{t-q} + z_t, t = q+1, ..., n$$

where $a = \begin{pmatrix} a_1 \\ a_2 \\ . \\ : \\ a_q \end{pmatrix} \in \mathbb{R}^q$ is the autoregression coefficients and $z_t$are independent and identically distributed random variables with mean 0 and variance σ^2. Moreover, we define the regression and autoregressive parameters jointly as $\theta = (\beta, a)$.

In 2007 Wang, Li and Tsai[**3**] suggested a method of solving LASSO problem in the context of regression with autoregressive errors. In order to shrink unnecessary coefficients to zero and get estimates of the parameters we have to minimize the lasso criterion

$$L_n(\theta) = \sum_{t=q+1}^{n} \left[(y_t - x_t^{'}\beta) - \sum_{i=1}^{q} a_i(y_{t-i} - x_{t-i}'\beta) \right]^2 + \lambda \sum_{i=1}^{p} |\beta_i| + \gamma \sum_{i=1}^{q} |a_i|, \tag{3.3.1}$$

where, λ and γ are shrinkage parameters (or, tuning parameters) for regression and autoregression coefficients respectively.

The equation contains both regression and autoregression parameters. So estimates of the coefficients can be calculated iteratively by solving the following equations :

$$\hat{\beta} = \sum_{t=q+1}^{n} \left[(y_t - x_t'\beta) - \sum_{i=1}^{q} a_i(y_{t-i} - x_{t-i}'\beta) \right]^2 + \lambda \sum_{i=1}^{p} |\beta_i| \text{ , for a fixed } a \tag{3.3.2}$$

$$\hat{a} = \sum_{t=q+1}^{n} \left[(y_t - x_t'\beta) - \sum_{i=1}^{q} a_i(y_{t-i} - x_{t-i}'\beta) \right]^2 + \gamma \sum_{i=1}^{q} |a_i| \text{ , for a fixed } \beta \tag{3.3.3}$$

So, by method of iteration we will get estimates of β and a. So, we start with a initial values of coefficients say a_0 and β_0, putting the initial value in (3.3.1) and (3.3.2) respectively and get updated values of the coefficients say β_1 and a_1. We repeat this process taking $\beta_1$and $a_1$as initial values if $|\sum_{i=1}^{p} |\beta_{0i}| - \sum_{i=1}^{p} |\beta_{1i}|| > 0.001$and get another updated values of the parameters. Finally we take $\hat{\beta} = \beta_{m+1}$and $\hat{a} = a_{m+1}$if $|\sum_{i=1}^{p} |\beta_{mi}| - \sum_{i=1}^{p} |\beta_{(m+1)i}|| < 0.001$.

3.4. Problem of using the approach

The autoregressive coefficients that comes out as lasso solution may be arbitrary which creates a problem in interpreting the model. Suppose, $q = 10$ and after applying lasso it comes out that $a_3 = a_6 = a_9 = a_{10} = 0$. Here $q = 10$ implies we assume our error are following autoregressive process of order 10 and after applying lasso as a result if we only $a_9 = a_{10} = 0$ then we can say that the error are following autoreressive process of order 8, not 10. But here we get $a_3 = 0, a_6 = 0$, that means 3rd and 6th autoregressive coefficients are not significant and which implies that observation on a particular time will depend on previous $1st, 2nd, 4th, 5th, 7th$ and $8th$ time points but not on previous 3rd and 6th time points. This interpretation doesnot carry a proper sense as the value of a observation of a particular time without depending on the previous $3rd$ observation, depends on previous $4th$ and $5th$ observations. So, we need to modify this approach slightly and inspite of using lasso in case of estimating autoregressive coefficients we should use ordered lasso.

3.5. Ordered LASSO

In 2014, Suo and Tibshirani [**5**] proposed an ordered version of the ℓ_1 regularized regression in the context of sparse time lagged regression. Suppose, we have reponse variable y and covariate information X. We consider a regression model and there should be natural ordering in the coefficients. So it is natural to propose an ordered constraint on the coefficients. Now if we apply lasso for variable selection and prediction purpose incorporating this ordered constraint the method is known as ordered lasso. The coefficients of ordered lasso is defined as

(3.5.1)

$$\hat{\beta}^{orderedlasso} = \arg\min \left\{ \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \sum_{j=1}^{p} |\beta_i| \right\} \text{ subject to } |\beta_1| \geq |\beta_2| \geq ... \geq |\beta_p|.$$

This setup makes sense when there is a natural ordering in the covariates. The main application of this approach is time-lagged regression, where we want to predict an outcome at time point t from the covariate information of previous q time points. However, this is not a convex problem so we cannot ensure that an optimal solution of the problem exists. Hence the problem is modified by writting $\beta_j = \beta_j^+ - \beta_j^-$ with $\beta_j^+, \beta_j^- \geq 0$. The following problem is proposed as

$$(3.5.2) \qquad minimize \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}(\beta_j^+ - \beta_j^-))^2 + \lambda \sum_{j=1}^{p} (\beta_j^+ - \beta_j^-) \right\}$$

subject to $\beta_j^+ \geq \beta_j^+ \geq ... \geq \beta_j^+ \geq 0$ and $\beta_j^- \geq \beta_j^- \geq ... \geq \beta_j^- \geq 0$. The use of positive and negative components makes the problem convex. So we can get an optimal solution to the problem. As a solution we get pairs $(\hat{\beta}_j^+, \hat{\beta}_j^-)$. Finally we get estimates of coefficients as $|\hat{\beta}_j| = \hat{\beta}_j^+ - \hat{\beta}_j^-$. This estimates are non-increasing in j.

Why this approach is a correction of previous approach ? Note that here we have an additional monotonicity constraint inclueded in our model. As we are interested in time-lagged data, due to that monotonicity constraint if any $\beta_j = 0$ at any stage of lasso then $\beta_{j+1} = \beta_{j+2} = ... = \beta_p = 0$. Now, suppose we work with

time-lag upto 10 and $\beta_6 = 0$, then we have $\beta_7 = ... = \beta_{10} = 0$. So that we can conclude that our data following autogressive process of order 5.

3.6. Our Methodology

Here in case of estimating autoregressive coefficients in 3.3.2, we consider the problem of lasso with an additional monotonicity constraint. The objective functions that we want to minimize to get estimates of regression and autoregression coefficients by iteration are given by,

$$\hat{\beta} = \sum_{t=q+1}^{n} \left[(y_t - x_t'\beta) - \sum_{i=1}^{q} a_i (y_{t-i} - x_{t-i}'\beta) \right]^2 + \lambda \sum_{i=1}^{p} |\beta_i| \text{ ,for a fixed } a \tag{3.6.1}$$

$$\hat{a} = \sum_{t=q+1}^{n} \left[(y_t - x_t'\beta) - \sum_{i=1}^{q} a_i (y_{t-i} - x_{t-i}'\beta) \right]^2 + \gamma \sum_{i=1}^{q} |a_i|, \text{ subject to } |a_1| \geq |a_2| \geq ... \geq |a_q| \text{ for a fixed } \beta \tag{3.6.2}$$

This set up makes sense as there is a natural ordering in time. Here also we get the estimates of the parameters by mehod of iteration solving equations (3.6.1) and (3.6.2) simultaneously as mentioned in section(3.3).

CHAPTER 4

Data Analysis and Results

We want to predict the "ozone" layer of Los Angeles based on some meterological measurements and the data was collected by Leo Breiman in 1976. Here our response variable is "ozone" and others are covariates. We want to regress "ozone" on other covariates. But the data is collected on daily basis and all the variables that are measured depends on time. Here we want to use LASSO for variable selection and prediction. But due to time dependent structure in the data autoregressive errors are assumed.

We start with the scaled data which is usual practice as different varibles are in different scale so due do scaling variables becomes unit free with zero mean and unit variance. Then we fit classical linear regression in the data where response variable y is "ozone" and the other variables create the matrix of covariates,say X. The estimates of the coefficients of the model is given in Table 1.

TABLE 1. *Regression coefficients for OLS*

vh	*wind*	*humidity*	*temp*	*ibh*	*dpg*	*ibt*	*vis*	*doy*
-0.0677	*-0.0057*	*0.1996*	*0.4951*	*-0.0562*	*-0.0165*	*0.2801*	*-0.0799*	*-0.1171*

From the above table we can see that coefficients corrosponding to some variables are very small so we can shrink the coefficients using lasso.

We take a large sequence of λ (the shrinkage parameter of regression coefficients), from 0 to 100 of length 1000 in each case.

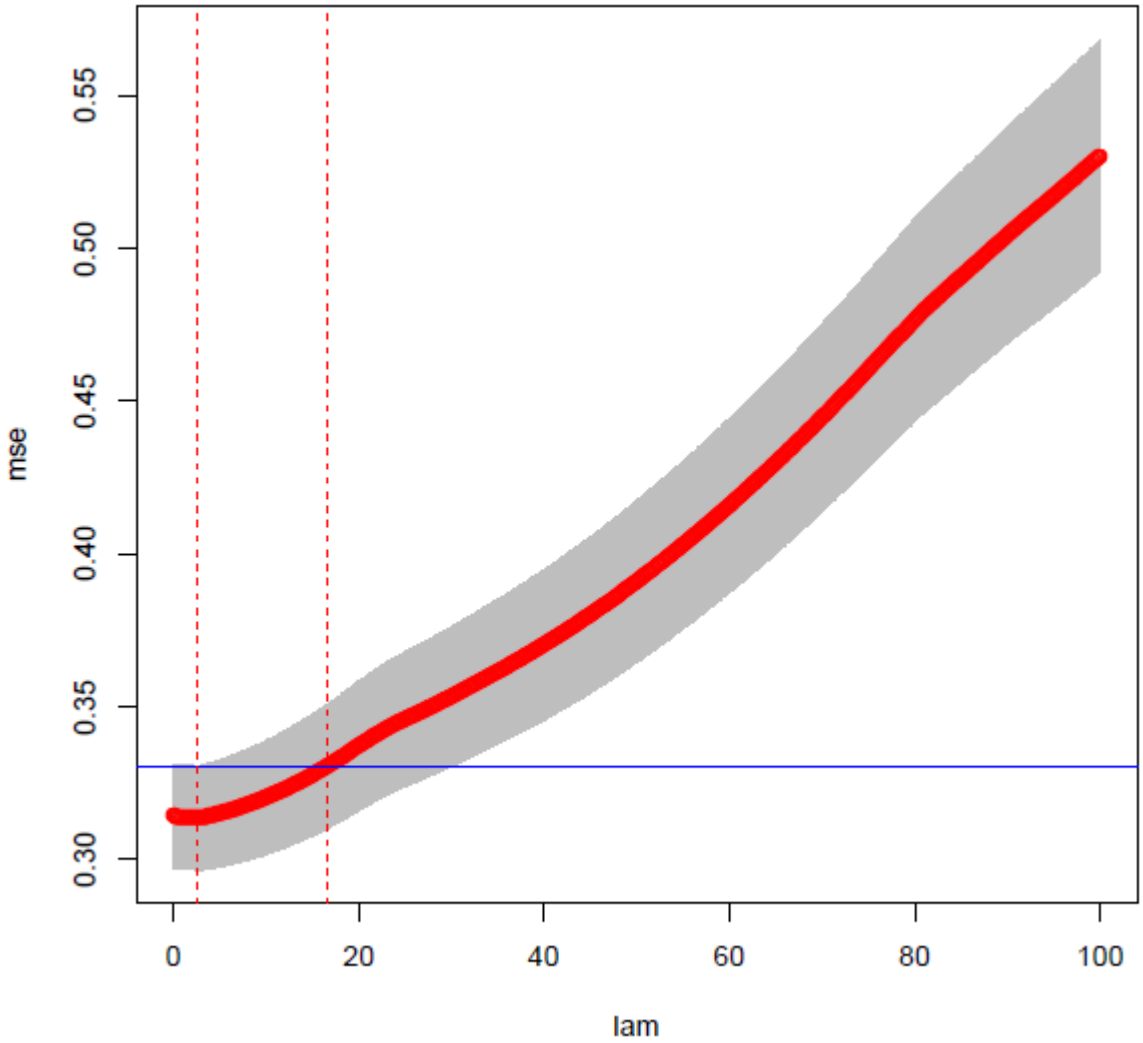
We consider that our data have autoregressive errors of order 1. We take initial value of autoregressive parameter $\rho_0 = 0.7$ and regression coefficients$\beta_0 = (0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95)^{'}$ (arbitrarily).Then we run the iteration procedure is mentioned in (3.2) using lasso to estimate regression coefficients and classical linear regression to estimate autoregressive coefficient. As the number of observations is very less so we do 10 folds cross-validation to calculate the prediction error and optimal choice of shrinkage parameter. Finally after 12 iteration steps we have :

$\hat{\rho} = 0.2739$ and the estimates of regression coefficients $\hat{\beta}$ given in Table2 with the cross-validation prediction error 0.3731 and optimal choice of $\hat{\lambda} = 16.6166$ in (4.0.1)

TABLE 2. *Regressive coefficients while error process is AR 1*

vh	*wind*	*humidity*	*temp*	*ibh*	*dpg*	*ibt*	*vis*	*doy*
0.000	*0.0039*	*0.1837*	*0.4760*	*-0.1026*	*0.000*	*0.1881*	*-0.0669*	*-0.0947*

FIGURE 4.0.1. Lasso path while errors are following AR 1 process



The graph represents the cross-validation MSE curve with one-standard error rule. Middle bolded red curve is MSE curve and intersction of blue horizontal line and MSE represents the optimal choice of λ and first red vertical line repersents the choice of λ where MSE is minimum

But the error may follow autoregressive process of order $q(\geq 2)$. So we consider a autoregressive process of order 10 taking intial value of autoregressive coefficients as $a_0 = (0.92, 0.78, 0.32, 0.66, 0.74, 0.31, 0.75, 0.99, 0.53, 0.76)'$ and $\beta_0$as before (arbitrarily). Then we run the iteration process as described in (3.3) using lasso to estimate both the regression and autoregression parameters. After 15 iteration

we have the estimates of regression coefficients $\hat{\beta}$ in Table 3 and the estimates of autoregression coefficients $\hat{a}$ in Table 4.

TABLE 3. *Regression coefficients when error process is AR* 10

vh	*wind*	*humidity*	*temp*	*ibh*	*dpg*	*ibt*	*vis*	*doy*
0.000	*0.000*	*0.1417*	*0.4502*	*-0.1134*	*0.000*	*0.1418*	*-0.0178*	*0.000*

TABLE 4. *autoregressive coefficients of AR 10 process*

a1	*a2*	*a3*	*a4*	*a5*	*a6*	*a7*	*a8*	*a9*	*a10*
0.2572	*0.0013*	*0.000*	*0.000*	*0.000*	*0.000*	*0.0316*	*0.0028*	*0.000*	*0.000*

From the above result we can see that the ozone layer of a particular day will depend of previous 1st,2nd,7th and 8th day, which is not interpretable. This is the problem that we have discussed in (3.4). So, to overcome this problem we should use ordered lasso in case of shrinkage estimation of autoregressive parameters and usual classical lasso in case of estiamtion of regression parameters.

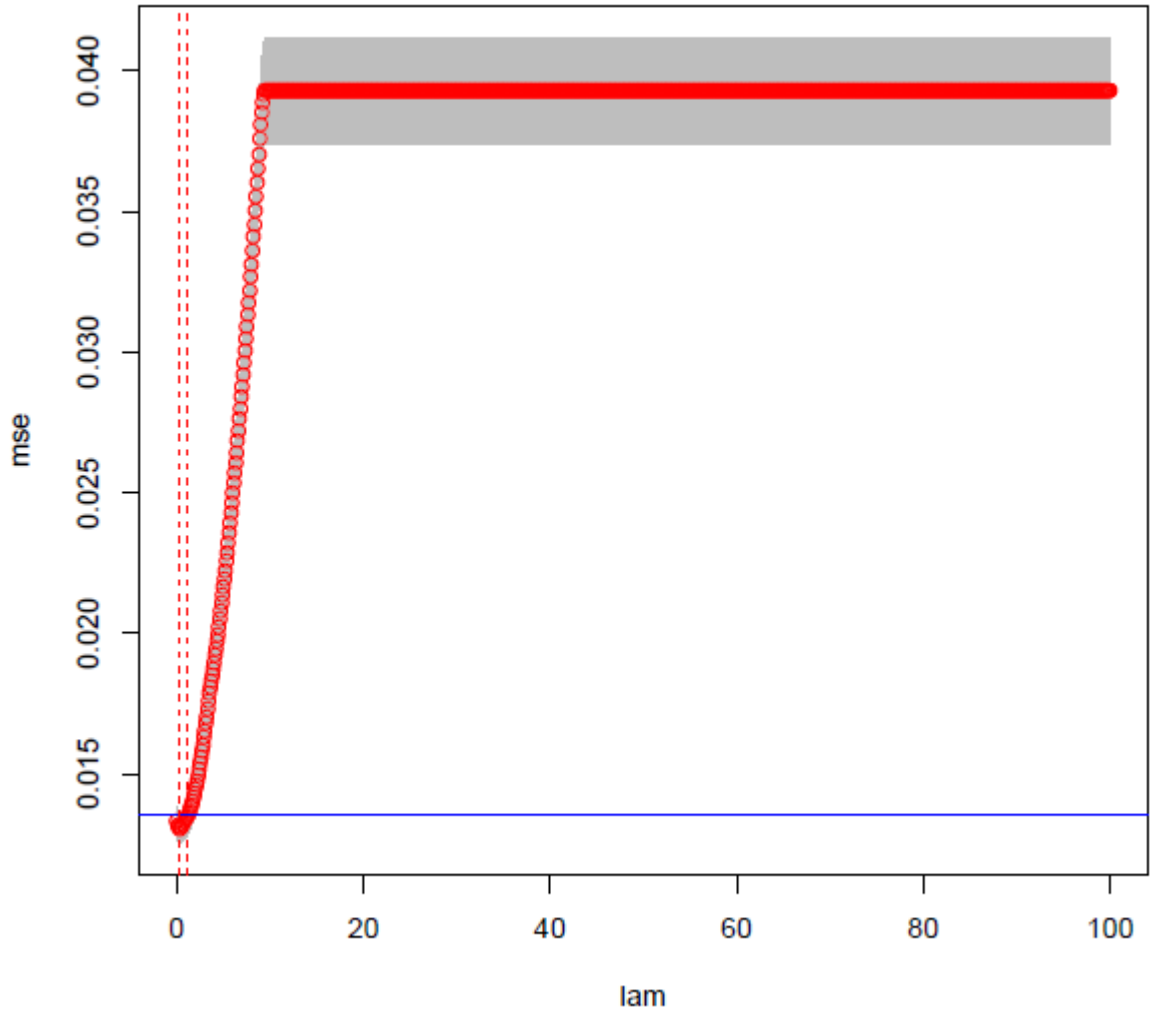
We run the iteration process discussed in (3.6) taking the initial values of a and β as before. Due to small size of data in hand we cannot split the dataset into train set and test set, so calculate 10 folds cross validation in case of estimating regression coefficients to calculate prediction error and optimal choice of shrinkage parameter $\lambda$and use inbuilt cv function of ordered lasso algorithm,which do 10 folds cross validation by default, to calculate optimal choice of ordered lasso shrinkage parameter γ. After 23 iterations we have the estimates of regression coefficients $\hat{\beta}$ in Table 5 and the estimates of autoregression coefficients $\hat{a}$ in Table 6, with cross-validation prediction error 0.0135 and optimal shrinkage parameters $\hat{\lambda} = 1.2012$ and $\hat{\gamma} = 5.4739$ (from Fig 4.0.2 and Fig 4.0.3)

So, we can say that our data have autoregressive errors of order 3.

TABLE 5. *Regression coeffficients when error process is AR* 10 *using lasso for ordered autoregressive coefficients*

vh	*wind*	*humidity*	*temp*	*ibh*	*dpg*	*ibt*	*vis*	*doy*
0.000	*0.000*	*0.19897*	*0.44638*	*-0.10689*	*0.000*	*0.21581*	*-0.0479*	*-0.06274*

FIGURE 4.0.2. Lasso path for AR 10 error process while AR coefficients are ordered

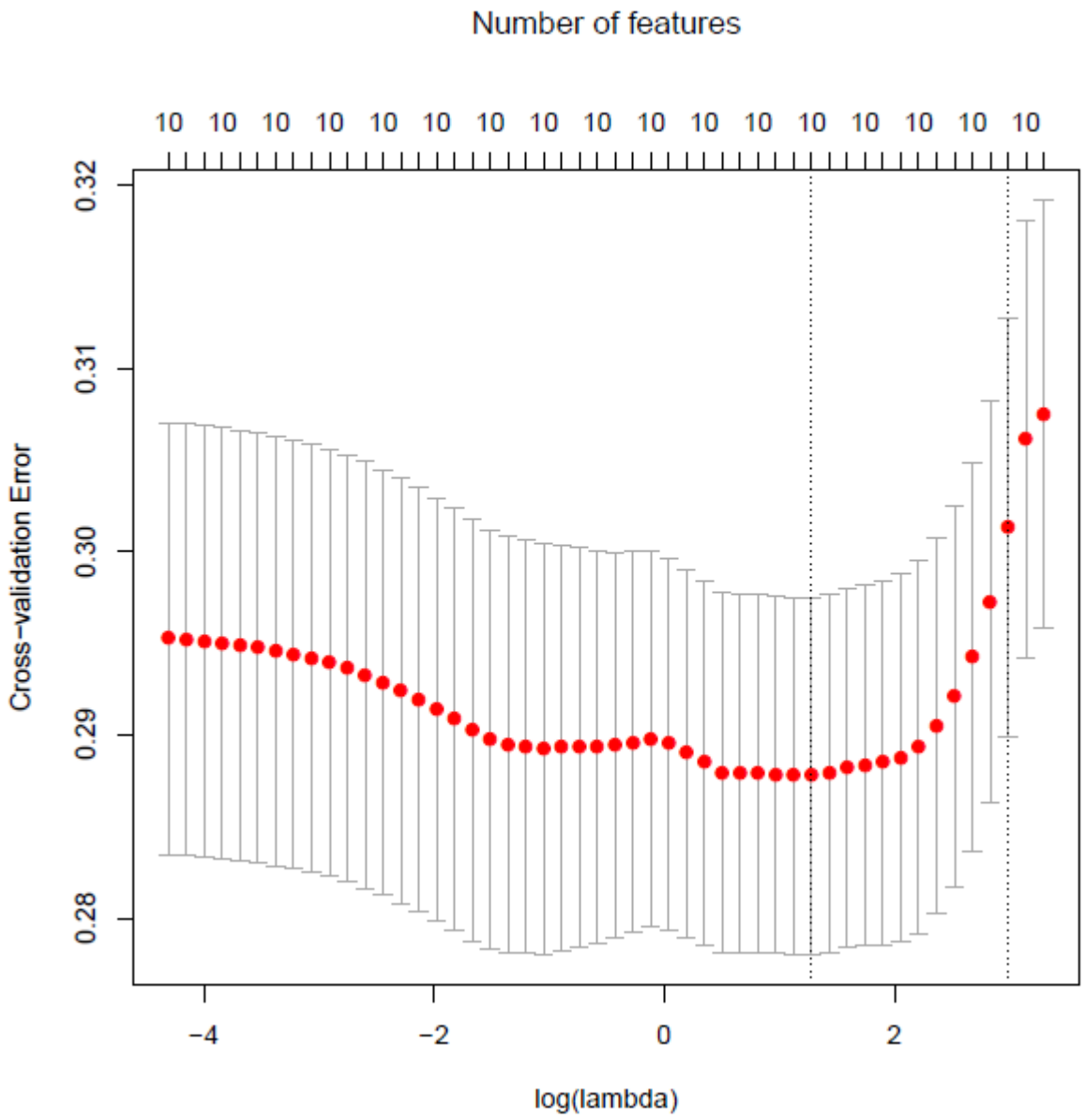


The graph represents the cross-validation MSE curve with one-standard error rule. Middle bolded red curve is MSE curve and intersction of blue horizontal line and MSE represents the optimal choice of λ and first red vertical line repersents the choice of λ where MSE is minimum

TABLE 6. *Autoregressive coefficients using ordered lasso*

a1	*a2*	*a3*	*a4*	*a5*	*a6*	*a7*	*a8*	*a9*	*a10*
0.2292	*0.02155*	*0.01554*	*0.000*	*0.000*	*0.000*	*0.000*	*0.000*	*0.000*	*0.000*

FIGURE 4.0.3. *Ordered lasso* path for error process of order 10



The graph represents the cross-validation MSE curve with one-standard error rule against log of shrinkage operator for error autoregressive process of order 10 and the optimal choice of γ is choosen where the MSE is minimum

Appendix

Dataset

The data was being presented by Leo Breiman who was a consultant on a project for which these data was been recorded. Here, the responses termed as "ozone", which is nothing but the log of the daily maximum of the hourly ozone layer depths in Upland California. Some data are missing so we have 330 observations in hand with 10 variables. The data that has been studied is available at,

"http://statweb.stanford.edu/ tibs/ElemStatLearn/data.html".

The variables details are following : (this destription is available on the same site)

"ozone : Upland Maximum Ozone"
"vh : Vandenberg 500 mb Height"
"wind : Wind Speed (mph)"
"humidity : Humidity (%)"
"temp : Sandburg AFB Temperature"
"ibh : Inversion Base Height"
"dpg : Daggot Pressure Gradient"
"ibt : Inversion Base Temperature"
"vis : Visibility (miles)"
"doy : Day of the Year"

```
data=read.table("C:\\Users\\SHWETA\\Desktop\\project\\LAozone.data",sep=",",header=T)
head(data)
```

```
##   ozone   vh wind humidity temp  ibh dpg ibt vis doy
## 1     3 5710    4       28   40 2693 -25  87 250   3
## 2     5 5700    3       37   45  590 -24 128 100   4
## 3     5 5760    3       51   54 1450  25 139  60   5
## 4     6 5720    4       69   35 1568  15 121  60   6
## 5     4 5790    6       19   45 2631 -33 123 100   7
## 6     4 5790    3       25   55  554 -28 182 250   8
```

R codes used

```
data=read.table("C:\\Users\\SHWETA\\Desktop\\project\\LAozone.data",sep=",",header=T)

attach(data)
data=scale(data)
y=data[,1]
x=data[,2:10]
```

```
x=as.matrix(x)

##... calssical multiple linear regression
lm(y~x+0)

##...lasso

require(lars)

beta0=rep(0.95,9)
rho=0.7;n=length(y)

P=diag(1,n)
P[1,1]=sqrt(1-rho*rho)
for(i in 2:n)
{
P[i-1,i]=-rho
}

y1=P%*%y
x1=P%*%x

##...function for lasso and CV

lam=seq(0,100,length=1000)
k=10
n=nrow(x); nlam=length(lam)
s=floor(n/k); h=1:n
fit.lasso=array(0,dim=c(k,nlam,s))
error.lasso=matrix(0,nrow=k,ncol=nlam)
cv.lasso=matrix(0,nrow=k,ncol=nlam)
mse=NULL; sd=NULL; UB=NULL; LB=NULL
for(i in 1:k)
{
folds=split(sample(1:n),ceiling(seq_along(sample(1:n))/s))
train.y=y[-folds[[i]]]; train.x=x[-folds[[i]],]
test.y=y[folds[[i]]]; test.x=x[folds[[i]],]
model=lars(train.x,train.y,normalize=F,type="lasso",intercept=F)
b=coef.lars(model,s=lam,mode="lambda")
fit.lasso[i,,]=b%*%t(test.x)
cv.lasso[i,]=rowMeans(scale(fit.lasso[i,,],center=test.y,scale=F)^2)
error.lasso[i,]=s*cv.lasso[i,]
}

mse=colMeans(cv.lasso)
sd=apply(cv.lasso,2,sd)/sqrt(k)
UB=mse+sd
LB=mse-sd
```

```
a=which.min(mse)
l.hat=lam[a]
output=list("lam.cv"=l.hat)
t=which(mse<UB[a])
opt.lam=lam[max(t)]
p.error=mse[max(t)]
output=list(c("lam.cv"=l.hat,"lam.opt"=opt.lam,"prediction.error"=p.error))
plot(lam,mse,type="b",ylim=c(min(LB),max(UB)),col="red")
for(i in 1:nlam)
{
segments(lam[i],UB[i],lam[i],LB[i],col="gray")
}
lines(lam,mse,col="red",type="b")
abline(v=opt.lam,lty=2,col="red")
abline(h=UB[a],untf=,col="blue")
a=abline(v=l.hat,lty=2,col="red")
beta.hat=coef.lars(model,s=l.hat,mode="lambda")
output=list(c("lam.cv"=l.hat,"lam.opt"=opt.lam,"prediction.error"=p.error),beta.hat)
beta1=output[[2]]

##...finding regression coefficient for ar1


z=x%*%beta1
z=y-z ;z1=z[2:n]; z2=z[1:n-1]
fit=lm(z1~z2-1)
rho1=coefficients(fit)
rho1
m=0

while( abs(sum(abs(beta0))-sum(abs(beta1)))>0.001 )
{
rho=rho1;beta0=beta1
P=diag(1,n)
P[1,1]=sqrt(1-rho*rho)
for(i in 2:n)
{
P[i-1,i]=-rho
}
n=nrow(x); nlam=length(lam)
s=floor(n/k); h=1:n
fit.lasso=array(0,dim=c(k,nlam,s))
error.lasso=matrix(0,nrow=k,ncol=nlam)
cv.lasso=matrix(0,nrow=k,ncol=nlam)
mse=NULL; sd=NULL; UB=NULL; LB=NULL
for(i in 1:k)
{
folds=split(sample(1:n),ceiling(seq_along(sample(1:n))/s))
```

```
train.y=y[-folds[[i]]]; train.x=x[-folds[[i]],]
test.y=y[folds[[i]]]; test.x=x[folds[[i]],]
model=lars(train.x,train.y,normalize=F,type="lasso",intercept=F)
b=coef.lars(model,s=lam,mode="lambda")
fit.lasso[i,,]=b%*%t(test.x)
cv.lasso[i,]=rowMeans(scale(fit.lasso[i,,],center=test.y,scale=F)^2)
error.lasso[i,]=s*cv.lasso[i,]
}

mse=colMeans(cv.lasso)
sd=apply(cv.lasso,2,sd)/sqrt(k)
UB=mse+sd
LB=mse-sd
a=which.min(mse)
l.hat=lam[a]
output=list("lam.cv"=l.hat)
t=which(mse<UB[a])
opt.lam=lam[max(t)]
p.error=mse[max(t)]
output=list(c("lam.cv"=l.hat,"lam.opt"=opt.lam,"prediction.error"=p.error))
plot(lam,mse,type="b",ylim=c(min(LB),max(UB)),col="red")
for(i in 1:nlam)
{
segments(lam[i],UB[i],lam[i],LB[i],col="gray")
}
lines(lam,mse,col="red",type="b")
abline(v=opt.lam,lty=2,col="red")
abline(h=UB[a],untf=,col="blue")
a=abline(v=l.hat,lty=2,col="red")
beta.hat=coef.lars(model,s=l.hat,mode="lambda")
output=list(c("lam.cv"=l.hat,"lam.opt"=opt.lam,"prediction.error"=p.error),beta.hat)
beta1=output[[2]]

z=x%*%beta1
z=y-z ;z1=z[2:n]; z2=z[1:n-1]
fit=lm(z1~z2-1)
rho1=coefficients(fit)
rho1
m=m+1
}
m;beta1;output;rho1

##....codes for AR q error process

fi0=c(0.92,0.78,0.32,0.66,0.74,0.31,0.75,0.99,0.53,0.76)

y.new=c()
x.new=NULL
```

```
for( i in 11:length(y))
{
y.1=sum(fi0*y[(i-10):(i-1)])
y.new=c(y.new,y.1)
x.1=fi0%*%x[(i-10):(i-1),]
x.new=rbind(x.new,x.1)
}

fit1=lars(x.new,y.new,normalize=F,type="lasso",intercept=F)
c.beta=coef(fit1)
cv.lasso.beta=cv.lars(x.new,y.new,type="lasso")
limit.beta=min(cv.lasso.beta$cv)+cv.lasso.beta$cv.error[which.min(cv.lasso.beta$cv)]
s.cv.beta=cv.lasso.beta$index[min(which(cv.lasso.beta$cv<limit.beta))]
beta1=coef(fit1,s=s.cv.beta,mode="fraction")
beta1


x.hat=x%*%beta0
e=y-x.hat
e.mat=NULL
l=length(fi0)+1
for(i in 1:l)
{
e.1=e[i:(319+i)]
e.mat=cbind(e.mat,e.1)
}

fit2=lars(e.mat[,1:10],e.mat[,11],intercept=FALSE,type="lasso",normalize=F)
c.fi=coef(fit2)
cv.lasso.fi=cv.lars(e.mat[,1:9],e.mat[,10],type="lasso")
limit.fi=min(cv.lasso.fi$cv)
s.cv.fi=cv.lasso.fi$index[which(cv.lasso.fi$cv==limit.fi)]
fi1=coef(fit2,s=s.cv.fi,mode="fraction")
fi1

m=0
while(abs(sum(abs(beta0))-sum(abs(beta1)))>0.001)
{
fi0=fi1;beta0=beta1
y.new=c()
x.new=NULL
for( i in 11:length(y))
{
a=(i-10):(i-1)
y.1=sum(fi0*y[a])
y.new=c(y.new,y.1)
x.1=fi0%*%x[a,]
```

```
x.new=rbind(x.new,x.1)
}
fit1=lars(x.new,y.new,normalize=F,type="lasso",intercept=F)
c.beta=coef(fit1)
cv.lasso.beta=cv.lars(x.new,y.new,type="lasso")
limit.beta=min(cv.lasso.beta$cv)+cv.lasso.beta$cv.error[which.min(cv.lasso.beta$cv)]
s.cv.beta=cv.lasso.beta$index[min(which(cv.lasso.beta$cv<limit.beta))]
beta1=coef(fit1,s=s.cv.beta,mode="fraction")

x.hat=x%*%beta0
e=y-x.hat
e.mat=NULL
l=length(fi0)+1
for(i in 1:l)
{
e.1=e[i:(319+i)]
e.mat=cbind(e.mat,e.1)
}
fit2=lars(e.mat[,1:10],e.mat[,11],intercept=FALSE,type="lasso",normalize=F)
c.fi=coef(fit2)
cv.lasso.fi=cv.lars(e.mat[,1:9],e.mat[,10],type="lasso")
limit.fi=min(cv.lasso.fi$cv)
s.cv.fi=cv.lasso.fi$index[which(cv.lasso.fi$cv==limit.fi)]
fi1=coef(fit2,s=s.cv.fi,mode="fraction")
m=m+1
}
m;beta1;fi1

##...codes for ordered lasso

require(orderedLasso)
y.new=c()
x.new=NULL

for( i in 11:length(y))
{
y.1=sum(fi0*y[(i-10):(i-1)])
y.new=c(y.new,y.1)
x.1=fi0%*%x[(i-10):(i-1),]
x.new=rbind(x.new,x.1)
}
lam=seq(0,100,length=1000)
k=10
n=nrow(x.new); nlam=length(lam)
s=floor(n/k); h=1:n
fit.lasso=array(0,dim=c(k,nlam,s))
error.lasso=matrix(0,nrow=k,ncol=nlam)
cv.lasso=matrix(0,nrow=k,ncol=nlam)
```

```
mse=NULL; sd=NULL; UB=NULL; LB=NULL
for(i in 1:k)
{
folds=split(sample(1:n),ceiling(seq_along(sample(1:n))/s))
train.y=y.new[-folds[[i]]]; train.x=x.new[-folds[[i]],]
test.y=y.new[folds[[i]]]; test.x=x.new[folds[[i]],]
model=lars(train.x,train.y,normalize=F,type="lasso",intercept=F)
b=coef.lars(model,s=lam,mode="lambda")
fit.lasso[i,,]=b%*%t(test.x)
cv.lasso[i,]=rowMeans(scale(fit.lasso[i,,],center=test.y,scale=F)^2)
error.lasso[i,]=s*cv.lasso[i,]
}
mse=colMeans(cv.lasso)
sd=apply(cv.lasso,2,sd)/sqrt(k)
UB=mse+sd
LB=mse-sd
a=which.min(mse)
l.hat=lam[a]
output=list("lam.cv"=l.hat)
t=which(mse<UB[a])
opt.lam=lam[max(t)]
p.error=mse[max(t)]
output=list(c("lam.cv"=l.hat,"lam.opt"=opt.lam,"prediction.error"=p.error))
plot(lam,mse,type="b",ylim=c(min(LB),max(UB)),col="red")

for(i in 1:nlam)
{
segments(lam[i],UB[i],lam[i],LB[i],col="gray")
}
lines(lam,mse,col="red",type="b")
abline(v=opt.lam,lty=2,col="red")
abline(h=UB[a],untf=F,col="blue")
a=abline(v=l.hat,lty=2,col="red")
beta.hat=coef.lars(model,s=l.hat,mode="lambda")
output=list(c("lam.cv"=l.hat,"lam.opt"=opt.lam,"prediction.error"=p.error),beta.hat)
beta1=output[[2]]
beta1

x.hat=x%*%beta0
e=y-x.hat
e.mat=NULL
l=length(fi0)+1
for(i in 1:l)
{
e.1=e[i:(319+i)]
e.mat=cbind(e.mat,e.1)
}
cv=orderedLasso.cv(e.mat[,10:1],e.mat[,11],intercept=F,standardize=F,trace=T,
```

```
method="Solve.QP",strongly.ordered=T)
plot(cv)
limit=min(cv$cv.err)+cv$cv.se[which.min(cv$cv.err)]
a=which(cv$cv.err<limit)
a=max(a)
s=cv$lamlist[a]
fi=orderedLasso(e.mat[,10:1],e.mat[,11],intercept=F,standardize=F,lambda=s,
strongly.ordered=T,method="Solve.QP")
fi1=fi$beta
fi1

m=0
while(abs(sum(abs(beta0))-sum(abs(beta1)))>0.001)
{
fi0=fi1;beta0=beta1
y.new=c()
x.new=NULL

for( i in 11:length(y))
{
y.1=sum(fi0*y[(i-10):(i-1)])
y.new=c(y.new,y.1)
x.1=fi0%*%x[(i-10):(i-1),]
x.new=rbind(x.new,x.1)
}

n=nrow(x.new); nlam=length(lam)
s=floor(n/k); h=1:n
fit.lasso=array(0,dim=c(k,nlam,s))
error.lasso=matrix(0,nrow=k,ncol=nlam)
cv.lasso=matrix(0,nrow=k,ncol=nlam)
mse=NULL; sd=NULL; UB=NULL; LB=NULL
for(i in 1:k)
{
folds=split(sample(1:n),ceiling(seq_along(sample(1:n))/s))
train.y=y.new[-folds[[i]]]; train.x=x.new[-folds[[i]],]
test.y=y.new[folds[[i]]]; test.x=x.new[folds[[i]],]
model=lars(train.x,train.y,normalize=F,type="lasso",intercept=F)
b=coef.lars(model,s=lam,mode="lambda")
fit.lasso[i,,]=b%*%t(test.x)
cv.lasso[i,]=rowMeans(scale(fit.lasso[i,,],center=test.y,scale=F)^2)
error.lasso[i,]=s*cv.lasso[i,]
}
mse=colMeans(cv.lasso)
sd=apply(cv.lasso,2,sd)/sqrt(k)
UB=mse+sd
LB=mse-sd
a=which.min(mse)
```

```
l.hat=lam[a]
output=list("lam.cv"=l.hat)
t=which(mse<UB[a])
opt.lam=lam[max(t)]
p.error=mse[max(t)]
output=list(c("lam.cv"=l.hat,"lam.opt"=opt.lam,"prediction.error"=p.error))
plot(lam,mse,type="b",ylim=c(min(LB),max(UB)),col="red")

for(i in 1:nlam)
{
segments(lam[i],UB[i],lam[i],LB[i],col="gray")
}
lines(lam,mse,col="red",type="b")
abline(v=opt.lam,lty=2,col="red")
abline(h=UB[a],untf=F,col="blue")
a=abline(v=l.hat,lty=2,col="red")
beta.hat=coef.lars(model,s=l.hat,mode="lambda")
output=list(c("lam.cv"=l.hat,"lam.opt"=opt.lam,"prediction.error"=p.error),beta.hat)
beta1=output[[2]]

x.hat=x%*%beta0
e=y-x.hat
e.mat=NULL
l=length(fi0)+1
for(i in 1:l)
{
e.1=e[i:(319+i)]
e.mat=cbind(e.mat,e.1)
}
cv=orderedLasso.cv(e.mat[,10:1],e.mat[,11],intercept=F,standardize=F,trace=T,
method="Solve.QP",strongly.ordered=T)
plot(cv)
a=min(cv$cv.err)
a=which(cv$cv.err==a)
s=cv$lamlist[a]
fi=orderedLasso(e.mat[,10:1],e.mat[,11],intercept=F,standardize=F,lambda=s,
strongly.ordered=T,method="Solve.QP")
fi1=fi$beta
m=m+1

}
m
beta1
output[[1]]
fi1
```

Acknowledgement

I would like to convey my sincere gratitude to my supervisor Prof. Atanu Kumar Ghosh, Department of Statistics, Presidency University, Kolkata, for his constant support, guidance and suggestions. I would also like to thank Prof. Suman Guha, Department of Statistics, Presidency University, Kolkata for his instantaneous help to complete this work. Last but not the least, I would also like to thank all my class fellows for their continuous support and encouragement.

Bibliography

[1] D. Cochrane and G.H. Orcutt. Application of least squares regression to relationships containing auto-correlated error terms. *American Statistical Association*, 1949.

[2] Trevor Hastie Robert Thibshirani Gareth James, Daniela Witten. *An Introduction to Statistical Learning*. Springer, 2013.

[3] Guodong Li Hansgheng Wang and Chih-Ling Tsai. Regression coefficient and autoregressive order shrinkage and selection via lasso. *Royal Statistical Society*, 2007.

[4] Richard A. Davis Peter J. Brockwell. *Introduction to Time Series and Forecasting*. Springer, 2002.

[5] Xiaotong Suo and Robert Tibshirani. An ordered lasso and sparse time-lagged regression. *Technometrics*, 2014.

[6] Robert Thibshirani. Regression shrinkage and selection via the lasso. *Royal Statisical Society*, 1996.

[7] Jerome Friedman Trevor Hastie, Robert Thibshirani. *The Elements of Statistical Learning*. Springer, 2017.