Dealing With Linear Regression Using Bayesian Elastic Net

PB19010390 Ergan Shang

Dec 2021

Abstract

A Bayesian elastic net approach is presented for variable selection and coefficient estimation in linear regression model. Using the method of Gibbs sampling, large amount of computation embarrassment can be avoided. Meanwhile, the penalty parameters can be chosen by an empirical Bayes method. And a real datase will be used to evaluate the approach presented.

Key words: Bayesian elastic net; tuning parameters; Monte Carlo EM algorithm; full conditional distribution

1 Data descriptions and goals

1.1 Source

The dataset used in this paper is PM10(some particles in the air which can be viewed as pollution) which is available in the R package **truncSP** reported by Lindemark and Karlsson(2011) [1]

1.2 Descriptions

The data used in this paper is the air pollution data measured by the Public Roads Administration in Norway. To be specific, a subsample consisting of

500 observations on 7 covariates plus an **outcome variable**, collected between October 2001 and August 2003 are used.Those 7 covariates are referring to the log(number of cars per hour)(x_1), temperatures at a height of two meters above the ground(x_2), wind speed in meters per second(x_3), the temperature difference between a height of 25 meters and a height of 2 meters above the ground(x_4), wind direction(x_5), time of day in hours(x_6) and day number(x_7). The outcome variable is hourly values of the log(concentration of NO2).

We list 10 rows as follows to make this example clear.

Table 1: A glimpse at data

PM10	cars	temp	wind.speed	temp.diff	wind.dir	hour	day
3.66356	7.74414	-4.4	4.2	0.0	18.0	19	116
3.04452	8.03398	-5.7	4.8	-0.3	69.1	9	506
3.71357	4.70048	-13.5	4.3	0.2	80.0	3	95
2.94444	7.52510	1.4	3.0	0.1	177.0	22	161
4.06044	7.76260	4.1	5.6	1.1	287.0	7	80
3.68888	7.88683	5.8	2.3	-0.1	200.0	9	33
3.33220	7.81521	2.7	1.9	0.4	228.0	7	129
3.36730	7.77779	7.1	8.9	0.2	220.0	15	155
2.07944	6.89163	4.1	2.0	0.1	183.0	9	132
1.94591	7.67740	1.1	5.2	0.1	43.1	10	480

1.3 Goals

By using Bayesian elastic net, we pursue the goal of achieving sparse and accurate estimation of regression coefficients. To make this accomplishment, we need to choose an appropriate Bayes prior with the parameters needed to be tuned. Then by calculating the full conditional distribution, we use R to run the Gibbs sampling leading to the trace plot and histogram of each regression coefficient.

2 Methods

2.1 A Bayesian hierarchical model

We first put forward the linear regression model as follows:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

where $\mathbf{y} = (y_1, \cdots, y_n)$ is the response observations of number n. $\mathbf{X}$ is an $n \times k$ matrix of covariates whose column number represents the number of factors whose corresponding coefficients are $\beta = (\beta_1, \cdots, \beta_k)$. $\epsilon = (\epsilon_1, \cdots, \epsilon_n)$ are independent with $\epsilon_i \sim N(0, \sigma^2)$. We are interested in estimating β and **identifying any nonzero elements**

Based on the Bayesian elastic net theory, we put forward the following prior:

$$\pi(\beta_j|\lambda_1, \lambda_2) = C(\lambda_1, \lambda_2)\frac{\lambda_1}{2}exp(-\lambda_1|\beta_j| - \lambda_2\beta_j^2)$$

where $C(\lambda_1, \lambda_2)$ are the normalizing constant. Under the assumption that β_j are independent variables, we can calculate the constant. Specifically:

$$1 = \int_{-\infty}^{+\infty} \pi(\beta_j|\lambda_1, \lambda_2)d\beta_j = \lambda_1 C(\lambda_1, \lambda_2)\int_0^{\infty} exp(-\lambda_1\beta_j - \lambda_2\beta_j^2)d\beta_j$$

$$= \lambda_1 C(\lambda_1, \lambda_2)\int_0^{\infty} exp(-\lambda_2(\beta_j + \frac{\lambda_1}{2\lambda_2})^2 + \frac{\lambda_1^2}{4\lambda_2})d\beta_j$$

Now change $z = \sqrt{\lambda_2(\beta_j + \frac{\lambda_1}{2\lambda_2})^2}$ we get

$$1 = \lambda_1 C(\lambda_1, \lambda_2)exp(\frac{\lambda_1^2}{2\lambda_2})\int_{\frac{\lambda_1}{2\sqrt{\lambda_2}}}^{\infty} exp(-z^2)dz$$

Let $z^2 = t$,we obtain

$$C(\lambda_1, \lambda_2) = (\frac{\lambda_1}{2\sqrt{\lambda_2}})^{-1}\frac{1}{\Gamma(\frac{1}{2}, \frac{\lambda_1^2}{4\lambda_2})}exp(-\frac{\lambda_1^2}{4\lambda_2})$$

where $\Gamma(\alpha, \beta)$ denotes the upper incomplete gamma function:$\Gamma(\alpha, \beta) = \int_\beta^\infty t^{\alpha-1}e^{-t}dt$
Now we get the Bayesian hierarchical model:

$$\mathbf{y}|\beta, \sigma^2 \sim N(\mathbf{X}\beta, \sigma^2\mathbf{I}_n)$$

$$\beta|\lambda_1, \lambda_2 \sim \Pi_{j=1}^k \pi(\beta_j|\lambda_1, \lambda_2)$$

$$\sigma^2 \sim 1/\sigma^2$$

$$\lambda_1 \propto \lambda_1^{a-1} exp(-b\lambda_2)$$

$$\lambda_2 \propto \lambda_2^{c-1} exp(-d\lambda_2)$$

2.2 Full conditional distributions and sampling

Now we make transformation of the prior distribution $\pi(\beta|\lambda_1, \lambda_2)$ as follows

$$\pi(\beta_j|\lambda_1, \lambda_2) = C(\lambda_1, \lambda_2)\frac{\lambda_1}{2} exp(-\lambda_1|\beta_j| - \lambda_2\beta_j^2)$$

$$= C(\lambda_1, \lambda_2)\frac{\lambda_1}{2} exp(-\lambda_2\beta_j^2) \int_{u_j>|\beta_j|} \lambda_1 exp(-\lambda_1 u_j)du_j = C(\lambda_1, \lambda_2)\frac{\lambda_1^2}{2} \int_{u_j>|\beta_j|} exp(-(\lambda_1 u_j + \lambda_2\beta_j^2))du_j$$

Using the results above, we propose the following hierarchical model with $\mathbf{u}$ which will make computation of full conditional distributions more effective

$$\mathbf{y}|\beta, \sigma^2 \sim N(\mathbf{X}\beta, \sigma^2\mathbf{I}_n)$$

$$\beta, \mathbf{u}|\lambda_1, \lambda_2 \sim \Pi_{j=1}^k C(\lambda_1, \lambda_2)\frac{\lambda_1^2}{2} exp(-(\lambda_1 u_j + \lambda_2\beta_j^2))$$

$$\sigma^2 \sim \frac{1}{\sigma^2}$$

$$\lambda_1 \sim \lambda_1^{a-1} exp(-b\lambda_1)$$

$$\lambda_2 \sim \lambda_2^{c-1} exp(-d\lambda_2)$$

Consequently, it is easy to calculate the following full conditional distributions:

$$\beta|\mathbf{y},\mathbf{X},\mathbf{u},\sigma^2,\lambda_1,\lambda_2 \sim N(\Sigma^{-1}\mathbf{X}^T\mathbf{y},\Sigma^{-1})\Pi_{j=1}^k \mathbf{1}_{|\beta_j|<u_j}$$

$$\mathbf{u}|\beta,\lambda_1 \sim \Pi_{j=1}^k Exp(\lambda_1)\mathbf{1}_{|\beta_j|<u_j}$$

$$\sigma^2|\mathbf{y},\mathbf{X},\beta \sim IG(\frac{n}{2},\frac{1}{2}||\mathbf{y}-\mathbf{X}\beta||^2)$$

where $\Sigma = \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} + 2\lambda_2\mathbf{I}_k$ and IG denotes the inverse gamma distribution. What should be noticed is that the sampling from $\mathbf{u}$ can be implemented as follows:

1. Draw $u_j^* \sim Exp(\lambda_1)$
2. Let $u_j = u_j^* + |\beta_j|$

2.3 Tuning parameters λ_1 and λ_2

We use Monte Carlo EM algorithm to fix the optimal parameters.

We treat $\lambda, \sigma^2, \mathbf{u}$ as missing data. The complete data log-likelihood is(under the condition $u_j > |\beta_j|$)

$$logp(\beta,\sigma^2,\mathbf{u},\lambda_1,\lambda_2|\mathbf{y},\mathbf{X}) \propto -klog(\frac{\lambda_1}{2\sqrt{\lambda_2}}\Gamma(\frac{1}{2},\frac{\lambda_1^2}{4\lambda_2})) - \frac{k\lambda_1^2}{4\lambda_2}$$

$$-\lambda_2(b+\sum_{j=1}^k u_j) + (a+2k-1)log\lambda_1 - \lambda_2(d+\sum_{j=1}^k \beta_j^2) + (c-1)log\lambda_2$$

Terms not involving λ_1, λ_2 are omitted. By using EM algorithm, we will encounter with the terms $\mathbb{E}^{\beta,\sigma^2,\mathbf{u}|\mathbf{y},\lambda^{(\ell-1)}}[\cdots]$ which are hard to calculate. We use the Monte Carlo EM algorithm proposed by [4] to deal with it. Let $\psi = (\lambda_1, \lambda_2)$, and have the iterations

$$\psi^{(k+1)} = arg\max_{\psi} \frac{1}{M}\sum_{k=1}^{M} logp(\psi|\mathbf{y},\mathbf{X},\mathbf{u},\sigma^2,\beta) \tag{2.1}$$

We sample $\beta, \mathbf{u}, \sigma^2$ from $\pi(\beta|\mathbf{u},\sigma^2,\psi), \pi(\mathbf{u}|\beta,\psi), \pi(\sigma^2|\beta)$ which is their full con-

ditional distributions with the initial value of $\psi^{(k)}$. To sum up, we have the following Monte Carlo EM algorithm:

1. set k=0 and initialize $\psi^{(0)} = (\lambda_1^{(0)}, \lambda_2^{(0)})$
2. for $j = 1, \cdots, M$ generate a sample $(\beta^{(j)}, \mathbf{u}^{(j)}, \sigma^{2(j)})$ from the Gibbs sampler with $\psi^{(k)}$
3. update $\psi^{(k)}$ using the expression (2.1)
4. at convergence of $\psi^{(k)}$ to $\hat{\psi}$ we can produce a final Gibbs sampler from the full conditional distributions

3 Implementation

Now we will use the method presented above to deal with the dataset mentioned. We choose the **posterior mean** to calculate each coefficient after dropping the burn in. At the same time, we will compare our method with other approaches such as least squares estimation, original Bayesian elastic net, lasso etc. Finally the trace plots and histograms for each coefficient will be provided.

Table 2: Estimation of each coefficient and residual

method	β_1	β_2	β_3	β_4	β_5	β_6	β_7	residuals
OLS	0.4863	-0.0088	-0.0877	0.0715	0.0005	-0.0077	0.0046	1508.0
glm_lasso	0.3037	0	-0.9464	0	0	0	0.0002	10029.4
glm_ridge	0.2700	0	-0.0743	0	0	0	0.000009	1628.2
glmnet	0.2952	0	-0.0910	0	0	0	0.0002	1347.3
EB_lasso	0.3155	0	-0.1007	0	0	0	0.0001	1235.8
lassoNEG	0	0	-0.0521	0	0	0.0243	0	5283.1
EBglmnet	0.3125	0	-0.1003	0	0	0	0.0001	1262.7
my method	0.4283	-0.1528	-0.6433	0	0.0934	0	-0.1741	1582.14

The second to fifth estimations are from the R package **glmnet** and the sixth to eighth estimations are from R package **EBglmnet**. Details can be obtained by the code below or just refer to the help document of the packages.

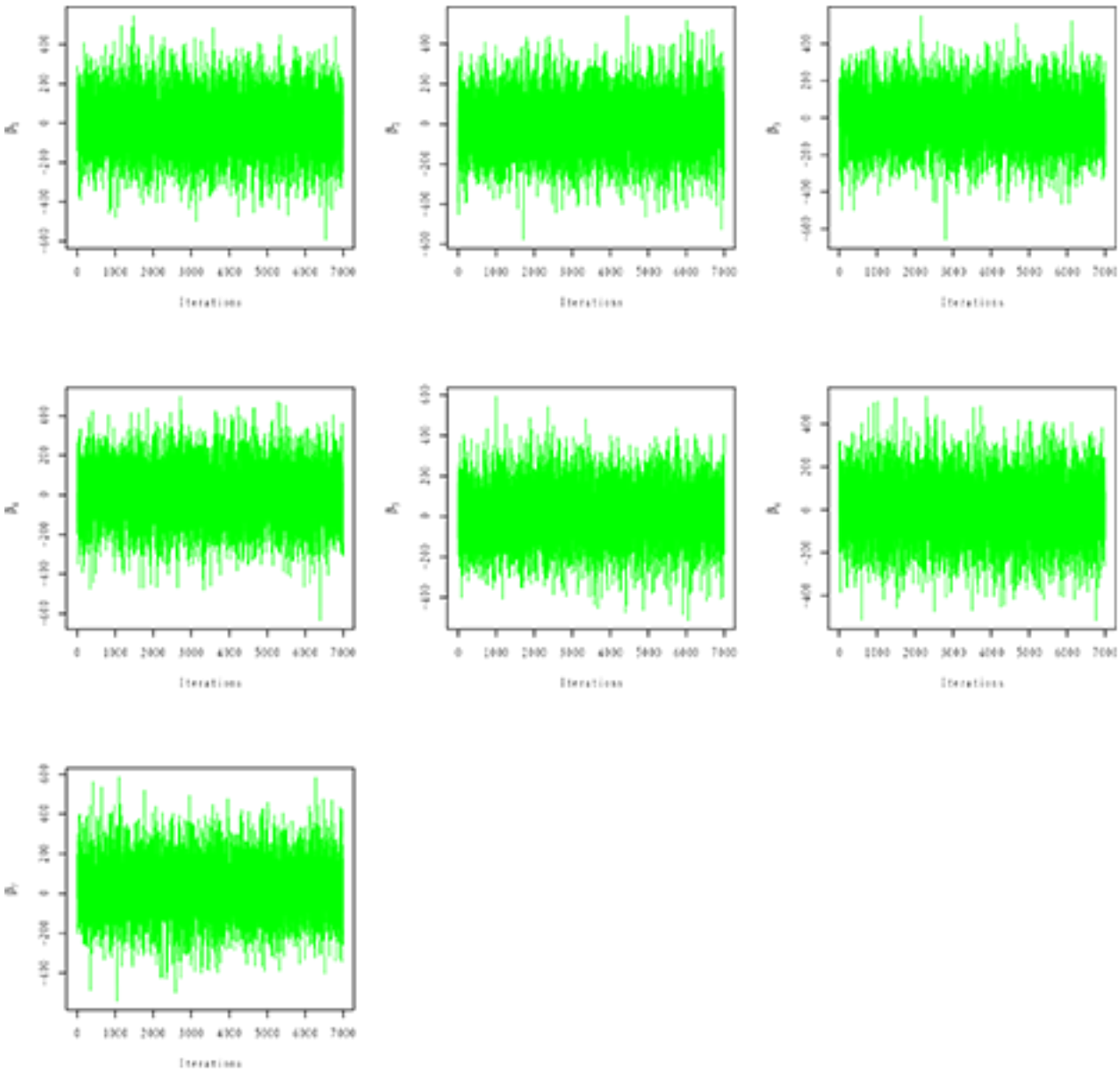

Figure 1: Trace plots of the regression coefficients

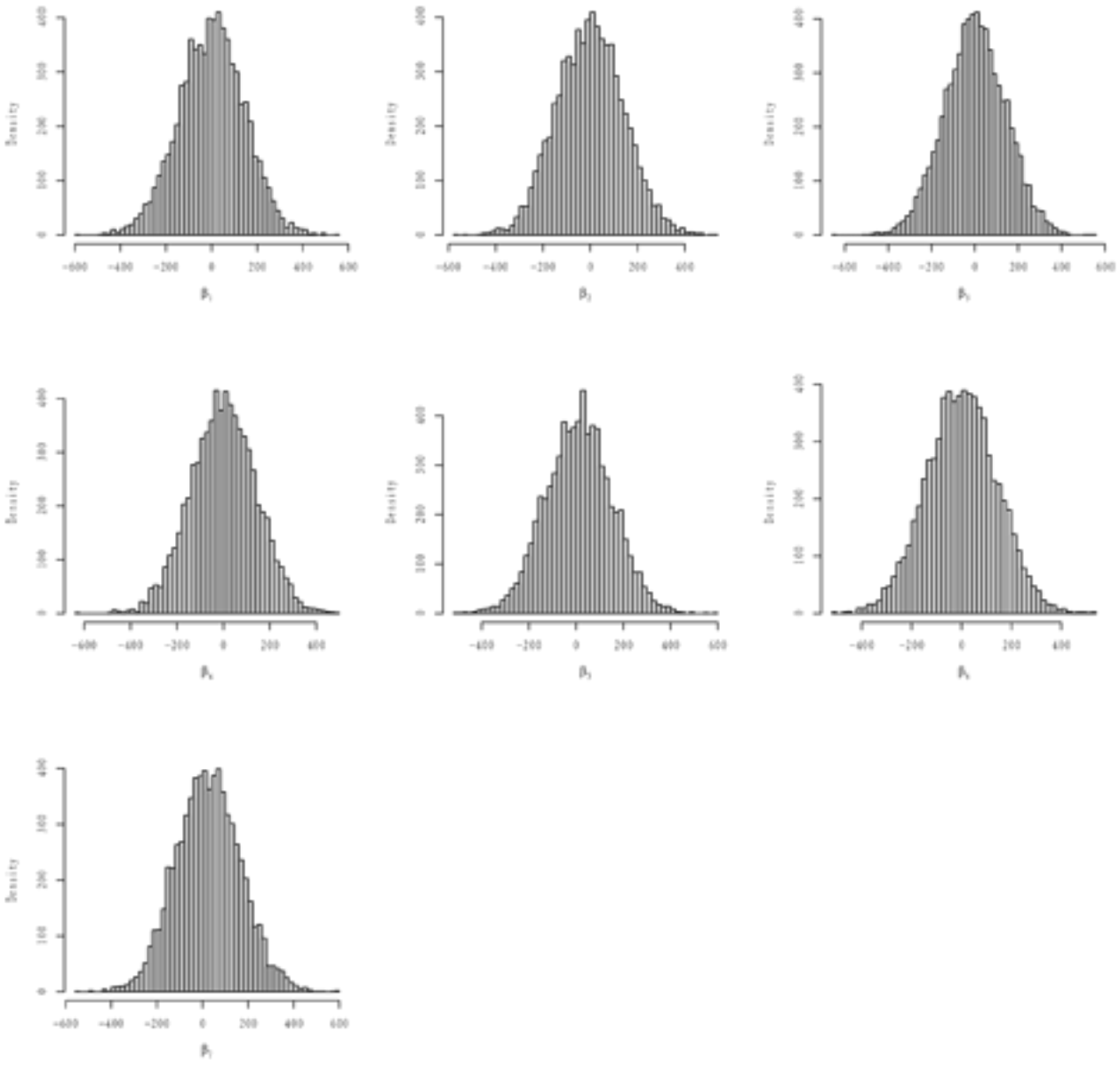

Figure 2: Histograms based on posterior samples

4 Conclusions and Evaluations

From the histograms above, we can reach the conclusion that the Markov chain **converges well** in this model. Meanwhile, the estimation of each coefficient is done at last. Compared with other priors and methods, our proposal has the ability to shrink some coefficients to zero but not all. The proposal elastic net has the ability to solve a regression problem with a lower residual.

However, it should be noted that our proposal did not shrink all irrelevant factors to zero. And it may have the potential to do a better job in terms of tuning parameters and selecting better priors. What's more, some **other numerical characteristics** can be selected to compute the regression coefficients which may improve the accuracy of our estimation. In the future, other better approaches for **determining tuning parameters** can be studied.

5 Code

```
#Input the data
data("PM10", package = "truncSP")
X<-as.matrix(PM10[,2:8])
y<-PM10[,1]
k=7;n=500
```

```
library(MCMCpack)
my_gamma<-function(t){
  return(t^(-1/2)*exp(-t))
}
logL<-function(a,b,c,d,lambda,m,k,para){
  logL<-0
  v<-integrate(my_gamma,lambda[1]^2/(4*lambda[2]),Inf)
  for(i in 1:m){
    logL<-logL-k*log(lambda[1]/(2*sqrt(lambda[2]))*v$value)-
    k*lambda[1]^2/(4*lambda[2])-lambda[1]*
```

```
    (b+sum(para[i,(k+1):(2*k)]))+
    (a+2*k-1)*log(lambda[1])-lambda[2]*(d+sum(para[i,1:k]^2))+
    (c-1)*log(lambda[2])
  }
  -logL
}

#do empirical Bayes
lambda10=1;lambda20=1
beta0<-rep(1,k);u0<-rep(2,k);sigma0<-1
eps<-.Machine$double.eps^0.25
m<-1000#set the length
Lambda<-numeric(2)
para<-matrix(0,m,2*k+1)#store the parameters:beta,u,sigma
Lambda<-c(lambda10,lambda20)#initial value
Lambdastar<-numeric(2)
para[1,]<-c(beta0,u0,sigma0)#initial value
u<-numeric(k)
a=1;b=1;c=1;d=1
while(abs(Lambda[1]-Lambdastar[1])>eps|
abs(Lambda[2]-Lambdastar[2])>eps){
  Lambdastar<-Lambda
  for(i in 2:m){
    Sigma<-1/para[i-1,2*k+1]*crossprod(X)+2*Lambdastar[2]*diag(k)
    mu<-solve(Sigma)%*%t(X)%*%y
    beta<-MASS::mvrnorm(1,mu,Sigma)
    #beta<-as.numeric(beta)
    para[i,1:k]<-beta
    for(j in 1:k){
      u[j]<-abs(beta[j])+rexp(1,Lambdastar[1])
    }
```

```
    para[i,(k+1):(2*k)]<-u
    sigma<-rinvgamma(1,n/2,0.5*sum((y-X%*%beta)^2))
    para[i,2*k+1]<-sigma
  }
  #para<-as.numeric(para)
  Lambda<-optim(c(5441,9991),logL,a=a,b=b,c=c,
  d=d,m=m,k=k,para=para)$par
  para[1,]<-para[m,]
  #use the results in the last iteration as the initial value
}
#Lambda is the tuning parameters
#now we get tuning parameter:lambda1&lambda2
```

```
#Now do the formal Gibbs sampling
m<-10000
realpara<-matrix(0,m,2*k+1)#parameters
realpara[1,]<-c(beta0,u0,sigma0)#initialize
u<-numeric(k)
for(i in 2:m){
  Sigma<-1/realpara[i-1,2*k+1]*crossprod(X)+2*Lambda[2]*diag(k)
  mu<-solve(Sigma)%*%t(X)%*%y
  beta<-MASS::mvrnorm(1,mu,Sigma)
  realpara[i,1:k]<-beta
  for(j in 1:k){
    u[j]<-abs(beta[j])+rexp(1,Lambda[1])
  }
  realpara[i,(k+1):(2*k)]<-u
  sigma<-rinvgamma(1,n/2,0.5*sum((y-X%*%beta)^2))
  realpara[i,2*k+1]<-sigma
}
realpara<-realpara[3000:m,]#throw off burn in
```

```
#trace plot
layout(matrix(c(1,2,3,4,5,6,7,8,9),3,3,byrow=TRUE))
plot(realpara[,1],xlab="Iterations",ylab=expression(beta[1]),
type="l",col="green")
plot(realpara[,2],xlab="Iterations",ylab=expression(beta[2]),
type="l",col="green")
plot(realpara[,3],xlab="Iterations",ylab=expression(beta[3]),
type="l",col="green")
plot(realpara[,4],xlab="Iterations",ylab=expression(beta[4]),
type="l",col="green")
plot(realpara[,5],xlab="Iterations",ylab=expression(beta[5]),
type="l",col="green")
plot(realpara[,6],xlab="Iterations",ylab=expression(beta[6]),
type="l",col="green")
plot(realpara[,7],xlab="Iterations",ylab=expression(beta[7]),
type="l",col="green")

#histogram
layout(matrix(c(1,2,3,4,5,6,7,8,9),3,3,byrow=TRUE))
hist(realpara[,1],breaks="scott",xlab=expression(beta[1]),
ylab="Density",main="")
hist(realpara[,2],breaks="scott",xlab=expression(beta[2]),
ylab="Density",main="")
hist(realpara[,3],breaks="scott",xlab=expression(beta[3]),
ylab="Density",main="")
hist(realpara[,4],breaks="scott",xlab=expression(beta[4]),
ylab="Density",main="")
hist(realpara[,5],breaks="scott",xlab=expression(beta[5]),
ylab="Density",main="")
hist(realpara[,6],breaks="scott",xlab=expression(beta[6]),
ylab="Density",main="")
```

```
hist(realpara[,7],breaks="scott",xlab=expression(beta[7]),
ylab="Density",main="")
```

```
#other methods
#OLS
beta_ols<-solve(t(X)%*%X)%*%t(X)%*%y
#use glmnet package
library(glmnet)
lambda_star<-cv.glmnet(X,y,alpha=1)$lambda.min
fit_lasso<-glmnet(X,y,alpha=1)#alpha=1 means lasso penalty
beta_lasso<-coef(fit_lasso,s=lambda_star)
lambda_star<-cv.glmnet(X,y,alpha=0)$lambda.min
fit_ridge<-glmnet(X,y,alpha=1)#alpha=0 means ridge estimator
beta_ridge<-coef(fit_ridge,s=lambda_star)
lambda_star<-cv.glmnet(X,y,alpha=0.5)$lambda.min
fit_ela<-glmnet(X,y,alpha=1/2)
#alpha=1/2 means an ordinary elastic net
beta_ela<-coef(fit_ela,s=lambda_star)
```

```
#use EBglmnet package which provides 3 different priors from us
library(EBglmnet)
hypara_lasso<-cv.EBglmnet(X,y,family="gaussian",
prior="lasso")$'optimal hyperparameter'
beta_EBlasso<-EBglmnet(X,y,family="gaussian",
prior="lasso",hyperparameters=hypara_lasso)
beta_EBlasso$fit

hypara_lassoNEG<-cv.EBglmnet(X,y,family="gaussian",
prior="lassoNEG")$'optimal hyperparameter'
beta_EBlassoNEG<-EBglmnet(X,y,family="gaussian",
prior="lassoNEG",hyperparameters=hypara_lassoNEG)
```

```
beta_EBlassoNEG$fit

hypara_ela<-cv.EBglmnet(X,y,family="gaussian",
prior="elastic net")$'optimal hyperparameter'
beta_EBela<-EBglmnet(X,y,family="gaussian",
prior="elastic net",hyperparameters=hypara_ela)
beta_EBela$fit
```

```
beta_ols<-c(0.4863,-0.0088,-0.0877,0.0715,0.0005,-0.0077,0.0046)
beta_glmlasso<-c(0.3037,0,-0.9464,0,0,0,0.0002)
beta_glmridge<-c(0.27,0,-0.0743,0,0,0,0.000009)
beta_glmnet<-c(0.2952,0,-0.0910,0,0,0,0.0002)
beta_EBlasso<-c(0.3155,0,-0.1007,0,0,0,0.0001)
beta_lassoNEG<-c(0,0,-0.0521,0,0,0.0243,0)
beta_EBglmnet<-c(0.3125,0,-0.1003,0,0,0,0.0001)
res1<-sum((y-X%*%beta_ols)^2)
res2<-sum((y-X%*%beta_glmlasso)^2)
res3<-sum((y-X%*%beta_glmridge)^2)
res4<-sum((y-X%*%beta_glmnet)^2)
res5<-sum((y-X%*%beta_EBlasso)^2)
res6<-sum((y-X%*%beta_lassoNEG)^2)
res7<-sum((y-X%*%beta_EBglmnet)^2)
beta_my<-c(mean(realpara[,1]),mean(realpara[,2]),
mean(realpara[,3]),mean(realpara[,4]),
mean(realpara[,5]),mean(realpara[,6]),mean(realpara[,7]))
res_my<-sum((y-X%*%beta_my)^2)
```

Bibliography

[1] Karlsson, M., & Lindmark, A. (2014). truncSP: An R Package for Estimation of Semi-Parametric Truncated Linear Regression Models. Journal of Statistical Software, 57(14), 1?19. https://doi.org/10.18637/jss.v057.i14

[2] Rahim Alhamzawi & Haithem Taha Mohammad Ali (2018) The Bayesian elastic net regression, Communications in Statistics - Simulation and Computation, 47:4, 1168-1178, DOI: 10.1080/03610918.2017.1307399

[3] Li Q , Lin N . The Bayesian elastic net[J]. Bayesian Analysis, 2010, 5(1):151-170.

[4] Casella G . Empirical Bayes Gibbs sampling[J]. Biostat, 2001.