ST443 GROUP PROJECT

Coordinate descent algorithm for solving the lasso problems

May 4, 2021

1 Introduction

The objective of this project is to apply coordinate descent algorithm on penalised regression problems such as the lasso (1) and the elastic net (2).

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

$$\frac{1}{2n}\sum_{i=1}^{n}(y_i-\sum_{j=1}^{p}x_{ij}\beta_j)^2+\lambda_1\sum_{j=1}^{p}|\beta_j|+\lambda_2\sum_{j=1}^{p}\beta_j^2. \qquad (2)$$

Algorithm 1 summarises the procedure for solving for the lasso and the elastic net problem. Further details can be seen in Friedman et al. (2007).

Apart from solving the lasso and the elastic net optimisation problems through the coordinate descent algorithm, we also attempt to address that the lasso has some practical limitations:

1. The lasso selects at most n variables before it saturates when p > n.

2. Assuming that we have a group of variables among which the pairwise correlation are very high, then the lasso has the tendency to select only one variable from the group and ignore the others.

3. Tibshirani (1996) empirically showed that the predictive performance of the ridge regression is more superior than the lasso when n > p.

To address the above limitations, we employ both l_1 and l_2 penalties in the form of (2). Furthermore, we run various simulation scenarios to demonstrate that the elastic net not only outperforms the lasso in terms of predictive accuracy, but also excels in variable selection.

The rest of the report would be divided into four parts: Part 2 provides the mathematical derivation of the lasso and the elastic net soft thresholding; Part 3 presents the pseudo code for both the lasso and the elastic net; Part 4 discusses the optimal selection of the regularisation parameters, e.g., $\hat{\lambda}$ in (1) and $\hat{\lambda}_1$, $\hat{\lambda}_2$ in (2); Part 5 shows the simulations results.

2 Soft Thresholding

2.1 Lasso

For l_1 penalty, the optimisation problem becomes $\frac{1}{2}(z-\theta)^2 + \lambda \mid \theta \mid$.

i) We have $\hat{\theta}$ = 0 if $\frac{1}{2}z^2 \leq \frac{1}{2}(z-\theta)^2 + \lambda\theta$, i.e. z ≤ $\frac{1}{2}\theta + \lambda$ for all $\theta > 0$ or $\frac{1}{2}z^2 \leq \frac{1}{2}(z-\theta)^2 - \lambda\theta$, i.e. $z \geq \frac{1}{2}\theta - \lambda$ for all $\theta < 0$. Hence $\hat{\theta}$ = 0 if -λ ≤ z ≤ λ.

ii) Otherwise, we will have $\hat{\theta}$ > 0 if we consider taking the first derivative of $\frac{1}{2}(z-\theta)^2$ w.r.t. θ, thus obtaining $\hat{\theta}$ - z + λ = 0, so $\hat{\theta}$ = z - λ > 0.

iii) Similarly, we will have $\hat{\theta}$ < 0 if we consider taking the first derivative of $\frac{1}{2}(z-\theta)^2 - \lambda\theta$ w.r.t. θ, thus obtaining $\hat{\theta}$ - z - λ = 0, so $\hat{\theta}$ = z + λ < 0.

Combining the above results, we obtain the soft thresholding rule $\hat{\theta}$ = sign(z) (| z | - λ) I(| z |> λ) for lasso.

2.2 Elastic Net

For the elastic net penalty, we could rewrite the resulting optimisation problem in terms of the lasso form.

$$\frac{1}{2}(z-\theta)^2 + \lambda_1\theta^2 + \lambda_2 \mid \theta \mid$$

$$= \frac{1}{2}\{(1+2\lambda_1)\theta^2 - 2\theta z + \frac{z^2}{1+2\lambda_1} + \frac{2\lambda_1 z^2}{1+2\lambda_1}\} + \lambda_2 \mid \theta \mid$$

$$= constant(\frac{2\lambda_1 z^2}{1+2\lambda 1}) + \frac{1}{2}(\frac{z}{\sqrt{1+2\lambda_1}} - \sqrt{1+2\lambda_1}\theta\})^2 + \frac{\lambda_2}{\sqrt{1+2\lambda_1}} \mid \sqrt{1+2\lambda_1}\theta \mid$$

$$= constant + \frac{1}{2}(\tilde{z}-\tilde{\theta})^2 + \tilde{\lambda} \mid \tilde{\theta} \mid,$$

11

where $\tilde{z} = \frac{z}{\sqrt{1+2\lambda_1}}$, $\tilde{\theta} = \sqrt{1+2\lambda_1}\theta$ and $\hat{\lambda} = \frac{\lambda_2}{\sqrt{1+2\lambda_1}}$. Applying the soft thresholding rule on $\hat{\theta}$, we have

$$\sqrt{1+2\lambda_1}\hat{\theta} = sign(\frac{z}{\sqrt{1+2\lambda_1}})(\frac{\mid z \mid - \lambda_2}{\sqrt{1+2\lambda_1}}) I(\frac{\mid z \mid}{\sqrt{1+2\lambda_1}} > \frac{\lambda_2}{\sqrt{1+2\lambda_1}}).$$

Hence the solution is $\hat{\theta}$ = $(1+2\lambda_1)^{-1}$ sign(z) (| z | - λ_2) I(| z |> λ_2).

3 Pseudo Code

Algorithm 1 Coordinate Descent Algorithm for Solving the Lasso and the Elastic Net

1 **Standardise X** and **y**.

2 **Initialise** $\beta_j = 0$, j = 1,...,p; t (tolerance) = 10^{-7}; J (cost function) = $\frac{1}{2n}\sum_{i=1}^{n}(y_i)^2$.

3. **Repeat** until convergence for j = 1,...,p; (convergence criteria: J_{diff} (difference in cost functions) < t).

(a) **Initialise** $J_{prev} = J^*$ (= J for the first iteration);

(b) **Compute** the partial residuals $r_{i,j} = y_i - \sum_{k\neq j} x_{i,j}\beta_k$;

(c) **Compute** the simple least squares coefficient of these residuals on the j-th predictor: $\beta_j^* = \frac{1}{n}\sum_{i=1}^{n} x_{i,j} r_{i,j}$;

(d) **Update** β_j by soft thresholding, see (2.1 for lasso and 2.2 for elastic net for details) and **Compute** J^*.

Lasso $\beta_j = \text{sign}(\beta_j^*)(|\beta_j^*| - \lambda)_+$,

$J^* = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$.

Elastic Net $\beta_j = (1 + 2\lambda_1)\text{sign}(\beta_j^*)(|\beta_j^*| - \lambda_2)_+$,

$J^* = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda_1\sum_{j=1}^{p}|\beta_j| + \lambda_2\sum_{j=1}^{p}\beta_j^2$.

where $(x)_+ = \max(x, 0)$.

(e) **Update** $J_{diff} = |J^* - J_{prev}|$.

4. **Return** $\beta_j\ (\frac{\sigma_y}{\sigma_X})$, j = 1,...,p.

4 Tuning Regularisation Parameters

We search for the optimal regularisation parameters λ for the lasso and λ_1 and λ_2 for the elastic net by using two approaches: the fixed validation set approach and the k-fold cross-validation. The optimal λ for the lasso and the and λ_1 and λ_2 for the elastic net are chosen by those resulting in the lowest MSE in the validation set.

The fixed validation set approach is a straightforward way which involves dividing the available set of observations into two parts, a training set and a validation set. This approach is conceptually simple, and is easy to implement. However, it suffers from a potential drawback: only a subset of the observations - those that are included in the training set rather than in the validation set - are used to fit the model. Empirical results show that statistical methods tend to perform worse when trained on fewer observations. Therefore, we also adopt k-fold cross-validation to address this potential drawback.

The k-fold cross-validation technique randomly divides the set of observations into k folds, of approximately equal size. One of these k folds is treated as the validation set, and the remaining k-1 folds are used as the training set. We then compute the mean squared error, MSE_1, on the observations on the held-out fold. This process is then repeated for k times; each time, a different group of observations is treated as a validation set. K different estimates of the test error, MSE_1, MSE_2, ..., MSE_k are generated in this procedure. The k-fold cross-validation is then computed by averaging these values,

$$CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

In this project, we perform k-fold cross-validation using k = 5, which has been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.

5 Simulation Results

In this section, we will use β of different sparcity, and vary n_{train}, n_{val}, σ, and the pairwise correlation in order to assess the performance of the lasso and the elastic using a validation set approach, as well as 5-fold cross validation. $\lambda = \lambda_1 = \lambda_2$ are 30 points, evenly spaced out in the region [0.01,2]. Whilst developing the coordinate descent algorithms, we observed that the parameters shrank significantly when λ, λ_1 and λ_2 increased past 2, thus we decided to set the upper bound of our set of possible values to 2.

Let's begin with $\beta = (3,1.5,0,0,2,0,0,0)$, $n_{train} = n_{val} = 20$, $n_{test} = 200$, $\sigma = 3$, and the corr(x_i, x_j) = $\rho^{|i-j|}$ where $\rho = [0,0.25,0.5,0.75,1]$. We repeat each simulation with 25 different data sets, and obtain the following results.

MSE(Standard error)				
ρ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
0	12.95(0.48)	12.67(0.5)	9.42(0.17)	8.89(0.16)
0.25	12.13(0.48)	11.99(0.46)	9.20(0.23)	8.70(0.21)
0.5	12.43(0.48)	11.96(0.4)	9.12(0.16)	8.59(0.15)
0.75	11.88(0.47)	11.38(0.4)	9.04(0.21)	8.58(0.2)
1	9.44(0.28)	9.42(0.28)	8.87(0.2)	8.78(0.2)

Non-Zero Coefficients				
ρ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
0	5.16	5.72	7.16	7.84
0.25	5.32	6.24	6.98	7.82
0.5	5.68	6.04	6.65	7.77
0.75	4.64	5.84	6.20	7.69
1	1.4	8	1.85	8

By examining the results, we can see that the elastic net outperforms the lasso in all cases by achieving a lower MSE. We can also see that the elastic net performs variable selection expect for the case where the variables are perfectly correlated. Furthermore, we can see that for highly correlated variables the lasso indeed selects one parameter and ignores the others, as the average number of non-zero coefficients for $\rho = 1$ was 1.4, where as the actual number was 3.

Now let's increase n_{train} and n_{val} and observe the results again. Let's try the values 50,150 and 500. The results we obtain are summarised in the tables below

MSE(Standard error)				
$n_{train} = n_{val}$	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
50	10.41(0.23)	10.29(0.23)	9.39(0.14)	8.98(0.13)
150	9.91(0.19)	9.91(0.2)	9.36(0.1)	9.13(0.11)
500	9.47(0.22)	9.47(0.22)	9.18(0.07)	9.1(0.07)

Non-Zero Coefficients				
$n_{train} = n_{val}$	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
50	5.84	6.52	6.74	7.73
150	6.48	6.16	6.21	7.52
500	5.76	5.48	5.52	7.52

The trend here is that as n_{train} and n_{val} increase, the MSE decreases meaning the prediction improves. We can see that the elastic net again outperforms the lasso as it achieves a lower MSE using both the validation set approach and the 5-fold CV.

Now let's vary σ and examine the results.

MSE(Standard error)				
σ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
1	1.53(0.07)	1.46(0.06)	1.04(0.02)	1(0.02)
30	32.15(1.19)	32.25(1.18)	24.72(0.57)	23.25(0.53)
10	116.16(3.66)	117.26(4.14)	101.36(2)	95.83(1.93)

Non-Zero Coefficients				
σ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
1	6.28	6	5.07	7.11
3	4.36	5.08	5.91	7.88
10	3.48	5.04	5.25	7.89

Here the elastic net outperforms the lasso in the 5-fold CV case, but and σ the data becomes noisier and noisier, so the algorithms begin to fail in predicting β accurately.

Now let's change β and run the same experimets to see if the results are similar. Let $\beta = (2,0,0,0,0,0,0,0)$. And again we begin by varying ρ.

MSE(Standard error)				
ρ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
0	11.14(0.48)	11.06(0.43)	9.27(0.23)	8.89(0.22)
0.25	10.36(0.27)	10.31(0.25)	8.97(0.13)	8.61(0.13)
0.5	9.96(0.27)	10.07(0.27)	8.88(0.13)	8.56(0.14)
0.75	10.65(0.34)	10.4(0.32)	9.13(0.15)	8.76(7.64)
1	9.69(0.21)	9.66(0.2)	9.17(0.14)	9.09(0.14)

Non-Zero Coefficients				
ρ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
0	3.16	3.52	3.18	7.82
0.25	3.56	4	2.22	7.73
0.5	3.24	3.72	2.26	7.71
0.75	2.6	3.72	3.06	7.62
1	2.16	8	2.02	8

Again, we see that the elastic net outperforms the lasso and in the case where the variables are not highly correlated, and it performs variable selection. 5-fold cross validation also produces lower MSE than the validation set approach.

MSE(Standard error)				
$n_{train} = n_{val}$	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
50	9.58(0.24)	9.58(0.24)	9.1(0.15)	8.84(0.14)
150	9.28(0.2)	9.29(0.2)	9.21(0.08)	9.02(0.08)
500	8.96(0.16)	8.94(0.16)	9.07(0.05)	8.98(0.05)

Non-Zero Coefficients				
$n_{train} = n_{val}$	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
50	3.12	3.6	2.48	7.62
150	4.52	4.56	3.71	7.67
500	4.24	3.96	5.42	7.3

As n_{train} and n_{val} increase, the MSE falls. The elastic net achieves lower MSE and in the validation set approach where n_{train}=n_{val} = 500, performs variable selection better than the lasso, as the average number of non-zero coefficients is closer to the actual number(3.96 vs 4.24, where the actual number is 1).

Elastic performs better than lasso, however with 5-fold CV it doesn't perform variable selection too well, but we can see that as n increase in starts to improve in that aspect too.

MSE(Standard error)				
σ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
1	1.24(0.04)	1.25(0.05)	1.07(0.02)	1.01(0.02)
3	27.61(0.58)	27.67(0.57)	25.03(0.49)	24.08(0.47)
10	109.71(1.92)	114.3(2.31)	104.08(2.06)	100.04(2.13)

Non-Zero Coefficients				
σ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
1	2.92	3.6	4.36	7.
3	2.44	3.24	2.86	7.86
10	1.928	3.76	2.4	7.82

Here again as σ increases the data becomes more affected by the noise, so both algorithms struggle to predict beta, however we can see that the elastic net provides the better performance, as the lowest MSE was achieved when 5-fold CV was applied with the elastic net.

Now we change β again, this time to (4,4,4,4,4,4,4,4) and we repeat the experiments.

MSE(Standard error)				
ρ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
0	16.9(1.37)	14.58(0.82)	9.36(0.2)	8.73(0.19)
0.25	18.08(1.22)	14.86(0.7)	9.65(0.15)	9.05(0.15)
0.5	15.77(0.82)	14.07(0.65)	9.79(0.17)	9.09(0.16)
0.75	15.69(1.15)	12.13(0.47)	9.38(0.2)	8.74(0.17)
1	9.57(0.35)	9.26(0.25)	9.05(0.18)	9.02(0.18)

Non-Zero Coefficients				
ρ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
0	8	8	8	8
0.25	8	8	8	8
0.5	8	8	8	8
0.75	7.89	8	8	8
1	1.64	8	1.79	8

In these cases we can see that the elastic net is superior. Not only that, but it's also not affected by highly correlated variables as we see in the case when $\rho = 1$. The actual number of non-zero coefficients is 8, but the lasso predicted on average 1.64 using the validation set approach and 1.79 using the 5-fold CV, which is quite far from the true value, where as the elastic net didn't encounter this problem.

Now let's increase n_{train} and n_{val}. The results obtained are summarised in the tables below.

MSE(Standard error)				
$n_{train} = n_{val}$	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
50	10.48(0.3)	10.2(0.26)	9.4(0.13)	8.9(0.12)
150	9.59(0.14)	9.43(0.14)	9.29(0.11)	8.99(0.1)
500	8.98(0.2)	8.87(0.19)	9.09(0.06)	8.98(0.06)

Non-Zero Coefficients				
$n_{train} = n_{val}$	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
50	8	8	8	8
150	8	8	8	8
500	8	8	8	8

Again, as n_{train} and n_{val} increase, the MSE for both the lasso and elastic net decreases. The elastic net achieves better performance than the lasso both using the validation set approach and 5-fold CV. Now let's vary σ again.

MSE(Standard error)				
σ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
1	2.18(0.14)	1.7(0.07)	1.16(0.02)	1.09(0.02)
3	43.25(2.08)	36.91(1.41)	26.81(0.4)	25.1(0.38)
10	148.4(3.6)	137.69(3)	103.79(1.92)	97.08(1.75)

Non-Zero Coefficients				
σ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
1	8	8	8	8
3	7.76	8	8	8
10	6.92	7.4	8	8

Observing the results, we can say that the elastic net, again, outperforms the lasso, even more so when σ gets large, the lasso using the validation set approach fails to select the correct number of non-zero coefficients.

Now let's consider a case where p > n. Suppose β is (2,0,0,0,0,0,0,0,0,0) repeated 25 times. We have $\sigma = 3$, $\text{corr}(x_i, x_j) = 0.5^{|i-j|}$, $n_{train} = n_{val} = 20$ and $n_{test} = 200$. We run 10 iterations and the results obtained can be found below.

MSE(Standard error)				
ρ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
0.5	105.55(4.75)	102.58(5)	17.48(0.45)	6.41(0.21)

Non-Zero coefficients				
ρ	Lasso Val Set	El net Val Set	Lasso 5-fold CV	El net 5-fold CV
0.5	5.6	119.5	108.54	187.26

Here, we see that the lasso using the validation set approach, selects on average on 5.6 coefficients, which is indeed less than $n_{train} = 20$.

6 Conclusion

In conclusion, based on the simulations we carried out, we can say that the elastic net is superior to the lasso as it tends to achieve lower MSE both using the validation set approach and the 5-fold cross validation approach. Moreover, the elastic net also performs variable selection and in the case where variables are highly correlated pairwise, it doesn't tend to select only one and ignore the rest. Overall, the 5-fold cross-validation achieves lower MSE compared with the fixed set validation approach. In the case of p > n, the lasso indeed selects less than n variables before it saturates; while in the case of n > p, the prediction performance of the lasso is dominated by the elastic net.

References

[1] Jerome Friedman et al. "Pathwise coordinate optimization". In: *The annals of applied statistics* 1.2 (2007), pp. 302–332.

[2] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

7 Appendix

R code

```
library(MASS)

# Coordinate descent algoritm for the lasso

lassoCorDes <- function(X,y,lambda){
  p <- ncol(X) # number of parameters
  n <- nrow(X) # number of observations
  #Normalize each column of X
  X <- apply(X,2,function(x) x - mean(x)) # centralize X
  X_sd <- apply(X,2,function(x) sqrt(sum(x^2)/n)) # sd of each X
  X <- apply(X,2,function(x) x/sqrt((sum(x^2)/n))) # standardize X
  y <- y - mean(y) # centralize y
  y_sd <- sqrt(sum(y^2)/n) # sd of y
  y <- y/y_sd # standardize y
  beta <- rep(0,p) # initialiaze vector beta containing beta_1,
     ..., beta_p
  tol <- 10^-7 # set convergence criterion
  J_dif <- 1 # initialize difference between cost function
  J <- sum(y^2)/(2*n) # initialize cost function
  iter <- 0 # iteration counter
  while(J_dif > tol){
    beta_prev <- beta
    J_prev <- J
    for (i in 1:p){
      res <- y - X[,-i]%*%beta[-i] # calculate residuals
      beta_star <- as.numeric((t(X[,i])%*%res)/n) # calculate beta
          star
      beta[i] <- sign(beta_star)*pmax(abs(beta_star) - lambda,0)
      # soft thresholding
    }
    J <- sum((y - X%*%beta)^2)/(2*n) + lambda*sum(abs(beta)) # new
        value
    # of cost function with the new estimate for beta
    J_dif <- abs(J - J_prev) # difference in cost functions
       between iterations
    iter <- iter + 1
```

```
    if (iter == 1000){
      return(beta*y_sd/X_sd) # Prevent function from entering
         infinite loop
      break
    }
  }
  return(beta*y_sd/X_sd) # scaling back beta
}

# Coordinate descent algoritm for the elastic net

elnetCorDes <- function(X,y,lambda1,lambda2){
  p <- ncol(X) # number of parameters
  n <- nrow(X) # number of observations
  #Normalize each column of X
  X <- apply(X,2,function(x) x - mean(x)) # centralize X
  X_sd <- apply(X,2,function(x) sqrt(sum(x^2)/n))
  X <- apply(X,2,function(x) x/sqrt((sum(x^2)/n))) # standardize X
  y <- y - mean(y) # centralize y
  y_sd <- sqrt(sum(y^2)/n) # sd of y
  y <- y/y_sd # standardize y
  lambda1 <- lambda1/y_sd
  lambda2 <- lambda2/y_sd
  beta <- rep(0,p) # initialiaze vector beta containing beta_1,
     ..., beta_p
  tol <- 10^-7 # set convergence criterion
  J_dif <- 1 # initialize difference between cost functions
  J <- sum(y^2)/(2*n) # initialize cost function
  iter <- 0
  while (J_dif > tol){
    beta_prev <- beta
    J_prev <- J
    for (i in 1:p){
      res <- y - X[,-i]%*%beta[-i] # calculate residuals
      beta_star <- as.numeric((t(X[,i])%*%res)/n) # calculate beta
          star
```

```
      beta[i] <- sign(beta_star)*pmax(abs(beta_star) - lambda1,0)/
         (1+lambda2)
      # soft thresholding
    }
    J <- sum((y - X%*%beta)^2)/(2*n) + lambda1*sum(abs(beta)) +
       lambda2*sum(beta^2)
    # new value of cost function with the new estimate for beta
    J_dif <- abs(J - J_prev) # difference in cost functions
       between iterations
    iter <- iter + 1
    if (iter == 1000){
      return(beta*y_sd/X_sd) # Prevent function from entering
         infinite loop
      break
    }
  }
  return(beta*y_sd/X_sd) # scaling back beta
}

# Selecting lambda for lasso and elastic net using validation set

lambdaVal <- function(X,y,n_train,n_val,n_test,lambda1,lambda2){
  n <- n_train + n_val + n_test
  n_rand <- sample(1:n,n) # shuffle n
  train <- n_rand[1:n_train] # training indices
  val <- n_rand[(n_train+1):(n_train+n_val)] # validation indices
  test <- n_rand[(n_train+n_val+1):n] # training indices
  X_train <- X[train,] # training predictors
  y_train <- y[train] # validation predictors
  X_val <- X[val,]
  y_val <- y[val]
  X_test <- X[test,]
  y_test <- y[test]
  lambda_lasso <- 0 # initialize the best lambda estimate
  lambda1_net <- 0 # initialize the best lambda1 estimate
  lambda2_net <- 0 # initialize the best lambda2 estimate
  beta_lasso_best <- 0
```

```r
beta_net_best <- 0
mse_val_min_lasso <- 10^20 # set initial validation set MSE to a
    very high value
mse_val_min_net <- 10^20
for(i in lambda1){
  beta_train_lasso <- lassoCorDes(X_train,y_train,i)
  # run coordinate descent on the training data
  mse_val_lasso <- mean((y_val - X_val%*%beta_train_lasso)^2)
  # calculate mse on validation set
  if(mse_val_lasso < mse_val_min_lasso){
    lambda_lasso <- i
    # update best lambda for lasso if mse on validation set is
       the lowest in the loop so far
    beta_lasso_best <- beta_train_lasso
    mse_val_min_lasso <- mse_val_lasso
    # replace the the previous minimum
  }
  for(j in lambda2){
    beta_train_net <- elnetCorDes(X_train,y_train,i,j)
    # beta obtained using coordinate descent on training data
    mse_val_net <- mean((y_val - X_val%*%beta_train_net)^2)
    # validation MSE with beta obtained from training data
    if(mse_val_net < mse_val_min_net){
      mse_val_min_net <- mse_val_net
      # update minimum validation mse
      beta_net_best <- beta_train_net
      # update best beta_estimate
      lambda1_net <- i
      # update best lambda1 for lasso if mse on validation set
      # is the lowest in the loop so far
      lambda2_net <- j
      # update best lambda2 for lasso if mse on validation set
      # is the lowest in the loop so far
    }
  }
}
```

```
  results <- list("MSE" = c(mean((y_test - X_test%*%beta_lasso_
     best)^2),
                            mean((y_test - X_test%*%beta_net_best)
                               ^2)),
                  "NonZeroCoefficients" = c(sum(beta_lasso_best !=
                      0),
                                            sum(beta_net_best !=
                                               0)))
  # put results into a list
  return(results)
}

lambdaCV <- function(X,y,n,lambda1,lambda2){
  folds = sample(rep(1:5, length = n)) # split indices into
     1,2,3,4 or 5
  lambda_lasso_cv <- 0 # initialize the best lambda estimate
  lambda1_net_cv <- 0
  lambda2_net_cv <- 0
  mse_cv_min_lasso <- 10^20 # set initial minimum MSE to a very
     high value
  mse_cv_min_net <- 10^20
  Nonzero_coefs_lasso_best <- 0
  # initialize the number of non-zero coefficients for the betas
     with the lowest mses
  Nonzero_coefs_net_best <- 0
  for(i in lambda1){
    mse_lasso <- 0 # initialize mse for each value of lambda
    Nonzero_coefs_lasso = c()
    for(fold in 1:5){
      beta_train_lasso <- lassoCorDes(X[folds != fold,], y[folds !
         = fold], i)
      # train on the rest of the folds
      mse_lasso <- mse_lasso +
        (length(y[folds == fold])/n)*mean((y[folds == fold] - X[
           folds == fold,]%*%beta_train_lasso)^2) # compute mse
      Nonzero_coefs_lasso = c(Nonzero_coefs_lasso, sum(beta_train_
         lasso != 0))
```

```
      # update number of non-zero coefficients
    }
    if(mse_lasso < mse_cv_min_lasso){
      mse_cv_min_lasso <- mse_lasso
      lambda_lasso_cv <- i # update best lambda
      Nonzero_coefs_lasso_best <- mean(Nonzero_coefs_lasso)
      # update average number on non-zero coefficients
    }
    for(j in lambda2){
      mse_net <- 0 # initialize mse for each combination of
         lambda1 and lambda2
      Nonzero_coefs_net = c()
      for(fold1 in 1:5){
        beta_train_net <- elnetCorDes(X[folds != fold1,], y[folds
           != fold1],i,j)
        # train on the rest of the folds
        mse_net <- mse_net +
          (length(y[folds == fold1])/n)*mean((y[folds == fold1] -
            X[folds == fold1,]%*%beta_train_lasso)^2)
        # compute mse
        Nonzero_coefs_net = c(Nonzero_coefs_net, sum(beta_train_
           net != 0))
        # update number of non-zero coefficients
      }
      if(mse_net < mse_cv_min_net){
        mse_cv_min_net <- mse_net
        lambda1_net_cv <- i # update best lambda1
        lambda2_net_cv <- j # update best lambda2
        Nonzero_coefs_net_best <- mean(Nonzero_coefs_net)
        # update average number on non-zero coefficients
      }
    }
  }
  results <- list("MSE" = c(mse_cv_min_lasso,mse_cv_min_net),
                  "NonZeroCoefficients" = c(Nonzero_coefs_lasso_
                     best,Nonzero_coefs_net_best))
  return(results)
```

```
}

# Function for MSE calculation

meanMSE <- function(true_beta,sig,n_train, n_val, n_test,
   iterations, lambda1, lambda2, corr){
  iter <- 0
  mse_val_lasso <- c()
  # initialize a vector of MSEs for the lasso using the validation
      set approach
  mse_val_net <- c()
  nonzero_coefs_lasso <- c()
  # initialize a vector with the average number of non-zero
     coefficients
  # for the lasso using the validation set approach
  nonzero_coefs_net <- c()
  mse_lasso_kcv <- c()
  # initialize a vector of MSEs for the lasso using 5-fold cross
     validation
  mse_net_kcv <- c()
  nonzero_coefs_lasso_kcv <- c()
  # initialize a vector with the average number of non-zero
     coefficients
  # for the lasso using 5-fold CV
  nonzero_coefs_net_kcv <- c()
  n <- n_train + n_val + n_test # number of observations
  p <- length(true_beta) # number of parameters
  Sigma <- matrix(0, ncol = p, nrow = p) # initialise covariance
     matrix
  for(i in 1:p){
    Sigma[i,i] <- 1
    for(j in 1:p){
      if(i != j){
        Sigma[i,j] <- corr^abs(i-j) # update covariance matrix
      }
    }
  }
```

```
while(iter < iterations){
  X <- mvrnorm(n, rep(0,p),Sigma) # generate X
  y <- X%*%true_beta + sig*rnorm(nrow(X)) # generate true values
  mseVal <- lambdaVal(X,y,n_train,n_val,n_test,lambda1,lambda2)
  mseCV <- lambdaCV(X,y,n,lambda1,lambda2)

  mse_val_lasso <- c(mse_val_lasso, mseVal$MSE[1])
  # update vector with lowest mse from validation set approach
  nonzero_coefs_lasso <- c(nonzero_coefs_lasso, mseVal$
     NonZeroCoefficients[1])
  # update vector with the number of non-zero coefficients
  # of beta that producest the lowest mse from validation set

  mse_val_net <- c(mse_val_net, mseVal$MSE[2])
  nonzero_coefs_net <- c(nonzero_coefs_net,mseVal$
     NonZeroCoefficients[2])

  mse_lasso_kcv <- c(mse_lasso_kcv, mseCV$MSE[1])
  # update vector with lowest mse from 5-fold cross validation
  mse_net_kcv <- c(mse_net_kcv, mseCV$MSE[2])
  nonzero_coefs_lasso_kcv <- c(nonzero_coefs_lasso_kcv, mseCV$
     NonZeroCoefficients[1])
  # update vector with the  average number of non-zero
     coefficients
  # from the betas that produces the lowest mse in 5-fold CV
  nonzero_coefs_net_kcv <- c(nonzero_coefs_net_kcv, mseCV$
     NonZeroCoefficients[2])
  iter <- iter + 1
}
results <- list("MSE" = c(mean(mse_val_lasso), mean(mse_val_net)
    ,
                          mean(mse_lasso_kcv),mean(mse_net_kcv))
                              ,
                "NonZeroCoefficients" = c(mean(nonzero_coefs_
                   lasso),
                                          mean(nonzero_coefs_net
                                             ),
```

```
                                                        mean(nonzero_coefs_
                                                           lasso_kcv),
                                                        mean(nonzero_coefs_net
                                                           _kcv)),
                          "StandardError" = c(sd(mse_val_lasso)/sqrt(
                             length(mse_val_lasso)),
                                                  sd(mse_val_net)/sqrt(length(
                                                     mse_val_net)),
                                                  sd(mse_lasso_kcv)/sqrt(
                                                     length(mse_lasso_kcv)),
                                                  sd(mse_net_kcv)/sqrt(length(
                                                     mse_net_kcv)) ))
  return(results)
}

#Trying different correlations

set.seed(443)
true_beta <- c(3,1.5,0,0,2,0,0,0)
lambda1 <- seq(0.01,2.0,length = 30)
lambda2 <- seq(0.01,2.0,length = 30)
corr <- seq(0,1,by=0.25)
results <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in corr){
  sim <- meanMSE(true_beta,3,20,20,200,25,lambda1,lambda2,i)
  results <- mapply(rbind,results,sim,SIMPLIFY = F)
}
results

# Increasing n

n_train <- c(50,150,500)
results1 <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in n_train){
  sim <- meanMSE(true_beta,3,i,i,200,25,lambda1,lambda2,0.5)
  results1 <- mapply(rbind,results1,sim,SIMPLIFY = F)
}
```

```
results1

# Varying sigma

sig <- c(1,5,10)
results2 <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in sig){
  sim <- meanMSE(true_beta,i,20,20,200,25,lambda1,lambda2,0.5)
  results2 <- mapply(rbind,results2,sim,SIMPLIFY = F)
}
results2
cat("################################################","\n")

# trying different correlations
true_beta <- c(2,rep(0,7))
results <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in corr){
  sim <- meanMSE(true_beta,3,20,20,200,25,lambda1,lambda2,i)
  results <- mapply(rbind,results,sim,SIMPLIFY = F)
}
results

# Increasing n

results1 <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in n_train){
  sim <- meanMSE(true_beta,3,i,i,200,25,lambda1,lambda2,0.5)
  results1 <- mapply(rbind,results1,sim,SIMPLIFY = F)
}
results1

# Varying sigma

results2 <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in sig){
  sim <- meanMSE(true_beta,i,20,20,200,25,lambda1,lambda2,0.5)
  results2 <- mapply(rbind,results2,sim,SIMPLIFY = F)
```

```
}
results2

cat("##############################################","\n")
true_beta <- rep(4,8)
results <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in corr){
  sim <- meanMSE(true_beta,3,20,20,200,25,lambda1,lambda2,i)
  results <- mapply(rbind,results,sim,SIMPLIFY = F)
}
results

# Increasing n

results1 <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in n_train){
  sim <- meanMSE(true_beta,3,i,i,200,25,lambda1,lambda2,0.5)
  results1 <- mapply(rbind,results1,sim,SIMPLIFY = F)
}
results1

# Varying sigma

results2 <- list("MSE", "NonZeroCoefficients", "StandardError")
for(i in sig){
  sim <- meanMSE(true_beta,i,20,20,200,25,lambda1,lambda2,0.5)
  results2 <- mapply(rbind,results2,sim,SIMPLIFY = F)
}
results2
cat("##############################################", "\n")

# Let's vary the number of parameter p

true_beta <- c(rep(c(2,rep(0,9)),25))
lambda1 <- seq(0.01,1,length = 5)
lambda2 <- seq(0.01,1,length = 5)
meanMSE(true_beta,3,20,20,200,10,lambda1,lambda2,0.5)
```