The Fussed Lasso

Sta315 Final Report

April 22,2022

Amrinder Sehmbi

1. Motivation

The lasso penalizes the least squares regression by the sum of the absolute values of the coefficients which allows sparse solutions such that many coefficients tend to equal. Robert Tibshirani and Michael Saunders propose the 'fused lasso' in their article titled *"Sparsity and Smoothness via the Fused Lasso"*, which is a generalization of lasso and fusion model that is designed for problems with features that have a meaningful ordering. The fused lasso penalizes the sum of the coefficients and their subsequent differences. This allows the sparsity of the coefficients and also sparsity of their differences. The fused lasso performs the best when the number of features p is much greater than the sample size, N. The motivation behind designing the fused lasso is to perform analysis on the prostate cancer data set which has a natural ordering of the features. Each sample i represents a blood serum which contains the intensity x_{ij} for many time-of-flight values t_j. The time of flight is related to the mass over charge ratio m/z of the proteins in the blood. There are 48 538 m/z sites with 157 healthy patients and 167 patients with cancer. The objective is to find m/z sites that help predict weather patient has cancer or does not have cancer.

2. Defining The Fussed Lasso

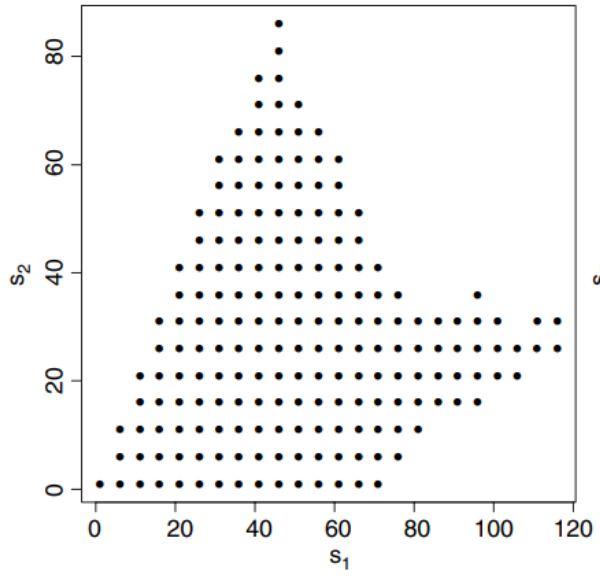
Let's define the prediction problem with N samples with the response $y_1, \dots, y_N$ and features $x_{ij}, i = 1, \dots, N, j = 1, \dots, p$. The response could be either quantitative or binary, representing two classes. We will focus on the case where p is much greater than N. The objective is to predict the response Y. This leads to the standard linear model $y_i = \sum_j x_{ij}\beta j + \varepsilon_i$ with error ε_i having a normal distribution with mean 0 and a constant variance. Since p is much larger than N, there needs to be some regularization or penalization of the feature to filter out unnecessary features for predicting the response. The authors considered many methods including ridge regression, partial least squares, and principal components regression. The lasso method was also originally considered because of its ability to produce sparse solutions. Lasso finds the coefficients $\hat{\beta} = (\hat{\beta}_1, \dots., \hat{\beta}_p)$ which satisfies $\hat{\beta} = argmin\{\sum_i (y_i - \sum_j x_{ij}\beta_j)^2\}$ subject to the constraint $\sum_j |\beta_j| \leq s$. The upper bound s is a tuning parameter such that for $s \to \infty$ we obtain many least square solutions for $p \gg N$ and for $s \to -\infty$ we obtain sparse solutions such that more coefficients tend to equal 0. Although lasso is attractive in the case where $p \gg N$, the main drawback of using lasso to analyze the prostate cancer data set in the article is that it ignores the ordering of the features. This issue led the authors to introduce the new method known as the fused lasso which satisfies $\hat{\beta} = argmin\{\sum_i (y_i - \sum_j x_{ij}\beta_j)^2\}$ subject to the constraints $\sum_j |\beta_j| \leq s\_1$ and $\sum_j |\beta_j - \beta_{j-1}| \leq s_2$. The first constraint allows sparsity in the coefficients and the second allows sparsity in their differences.

3. Computation

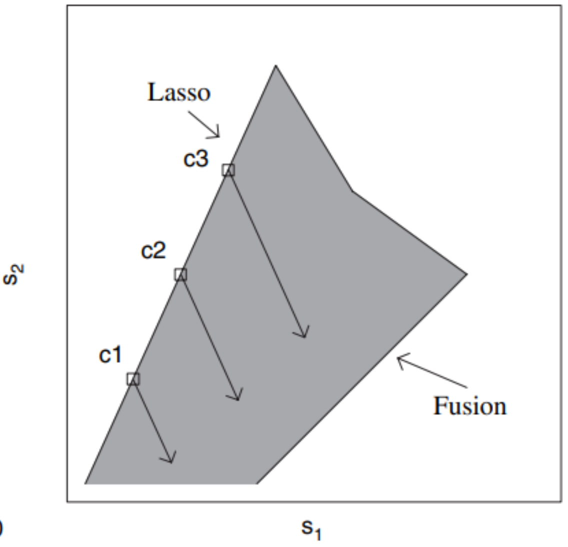
3.1 Fixed Bounds

There were two computational methods the author presented in the article. The first approach was starting with fixed values for the bounds s_1 and s_2. The criterion for the fused lasso defined above leads to quadratic programing. This problem is difficult to solve for large p, and could lead to storing p^2 elements in the database. This issue led the authors to use the algorithm SQOPT which is Sparse Quadratic Optimizer which is designed for quadratic programming problems with sparse linear constraints. This algorithm performs well with warm starts such that starting with fixed bounds s_1 and s_2, the optimal nearby values of these bounds can be found in reasonable time. For datasets where we have roughly 1000 predictors and 100 samples, the fixed approach works well.

3.2 Search Strategy



(a)



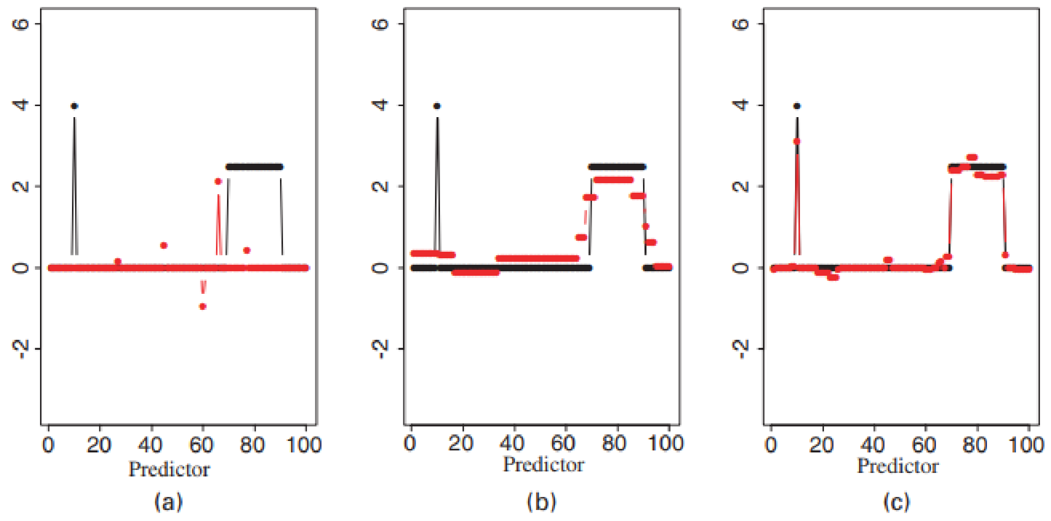
(b)

The second approach defined in the article is the search strategy which was used in the article’s simulation study, and we will be using a variation of this in our simulation. This strategy is best for when we have large datasets since we restrict the bounds as showcased in table (a) below, hence, we can’t use the fixed approach. It takes advantage of the (LARS) least angle regressions ability to efficiently compute the lasso which only penalizes the coefficients and fusion which only penalizes the successive differences of the coefficients. The LARS algorithm starts by normalizing all the values to have mean 0 and unit variance. Then we find a variable that is highly correlated to the errors. We move the regression line in the direction of this variable until we reach another variable that has higher correlation. If there are two variables that have the same correlation, move the regression line at an angle that is in between. This process is repeated until we have used all variables or until we a desired size model.
The fusion problem can be solved by transforming the data matrix X to $Z = XL^{-1}$ where L is a $p\ x\ p$ matrix with $L_{ii} = 1$ and $L_{i+1,i} = -1$ and $L_{ij} = 0$ with $\theta = L\beta$ and then transforming the data back after applying the LARS algorithm. To solve the fused lasso problem, we start by using LARS to retain the lasso bounds $(s_1(i), \infty)$, where $s_1(i)$ is the bound on the lasso constraint with i degrees of freedom. The optimal degrees of freedom can be found using cross validation. Next, we define the second bound of the

fused lasso to be $s_{2\max}\{s_1(\hat{\imath})\} = \sum_j |\beta_j\{s_1(\hat{\imath})\} - \beta_{j-1}\{s_1(\hat{\imath})\}|$, which is the largest of bound s_2 given s_1 which gives the optimal solution from the restricted set of bounds. This method is repeated twice more to compute the bounds s_1 and s_2 with $i/2$ and $(i + \min(N, p))/2$ degrees of freedom. The points $c_2 = [s_1(\hat{\imath}),\ s_{2\max}\{s_1(\hat{\imath})\}]$, $c_1 = [s_1(\hat{\imath}/2),\ s_{2\max}\{s_1(\hat{\imath}/2)\}]$ and $c_3 = [s_1((\hat{\imath} + \min(N,p))/2),\ s_{2\max}\{s_1((\hat{\imath} + \min(N,p))/2)\}]$ is plotted on the table (b). Once we have the points c_1, c_2, c_3, we fuse the solution by moving in the direction (1,-2) which was chosen by empirical experimentation which was not discussed in the article. We chose the set of bounds which is the furthest away from the pure fusion model which is represented by the lower bound in table (b). Finally, we then solve for the optimal coefficient which minimize the sum of squared residuals with subject to the chosen bounds.

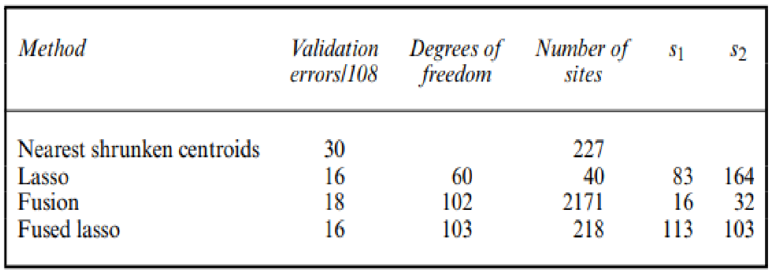
4. Advantages

4.1 $p \gg N$



As stated previously, the fused lasso performs the best when the number of predictors p is much greater than sample size N. The authors of the article conducted a mini simulation which illustrates the effect of the fused the lasso in the tables above. The data for the simulation consists of 100 predictors and 20 samples and was generated from the model $y_i = \sum_j x_{ij}\beta j + \ \varepsilon_i$ where the x_{ij} are from standard gaussians, ε_i is normally distributed with mean 0 and unit variance, sigma is equal to 0.05, bounds were chosen in each model to minimize the prediction error and there are two blocks of consecutive non-zero $\beta_j's$ shown by the black points in the tables above. Based on the results, the lasso seems to perform poorly compared to the fusion and fused lasso models. The fusion model seems to capture the plateau but does not clearly catch the peak value for the predictor valued 10. The fused lasso model seems to capture the true coefficients the best in general.

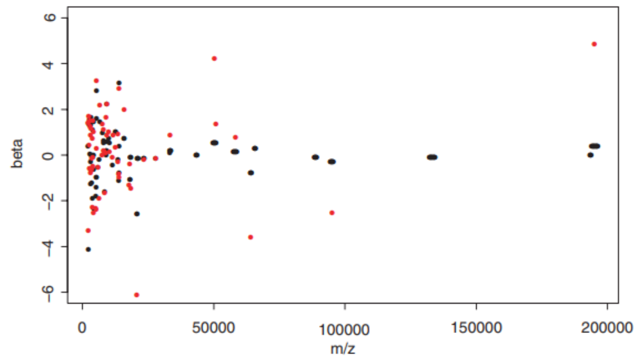
4.2 Ordered Features



Method	*Validation errors/108*	*Degrees of freedom*	*Number of sites*	s_1	s_2
Nearest shrunken centroids	30		227		
Lasso	16	60	40	83	164
Fusion	18	102	2171	16	32
Fused lasso	16	103	218	113	103

The existence of a meaningful order in the predictors of the data truly allows fused lasso to exhibit its prediction power. The authors illustrate this power by performing analysis on the prostate cancer data set which we described above. The training and validation sets were randomly created of size 216 and 108 patients respectively. The results are shown in the table above. We see that the fused lasso model has the lowest validation error along with the highest degrees of freedom and highest bounds for the constraints. The lasso also gives the same validation error, but it has less degrees of freedom and sets more coefficients equal to 0 compared to fused lasso. The fusion model gives a higher validation error with similar degrees of freedom, but it sets less coefficients to 0, compared to fused lasso. The fused lasso seems to find the balance between lasso and fusion model for all estimates when the features have a meaningful order.

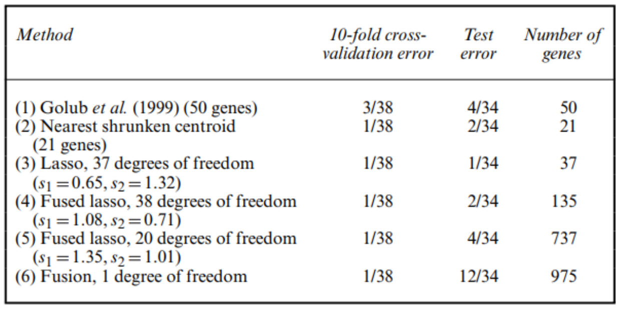
4.3 Sparsity



We know that lasso provides sparse solution subject to one constraint, but fused lasso penalized both the coefficient and their successive differences which allows sparsity of the coefficients and sparsity of their differences. In the case of $p \gg N$, sparsity of the lasso implies that we could have at most N non-zero coefficients and sparsity of the fused lasso implies that we could have at most N sequences of consecutive feature values with the same coefficient. From the prostate cancer data set from the article, we see the estimates of the coefficient using the lasso and fused lasso models in the graph above. The graph illustrates that the fused lasso sets non-zero coefficients for more m/z sites and it spreads out the coefficients more especially at higher values of the m/z sites, compared to lasso. The sparsity of the fused lasso allows use to consider the order of the features hence leading to more non-zero coefficients which lead to better prediction of the responses in the prostate cancer data set.

5. Disadvantages

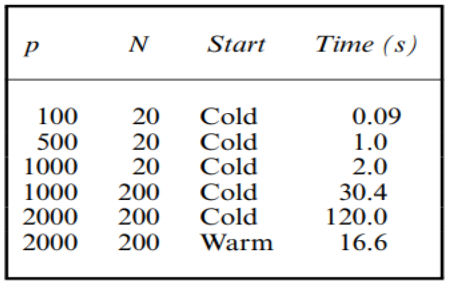
5.1 Unordered Features



Method	*10-fold cross-validation error*	*Test error*	*Number of genes*
(1) Golub *et al.* (1999) (50 genes)	3/38	4/34	50
(2) Nearest shrunken centroid (21 genes)	1/38	2/34	21
(3) Lasso, 37 degrees of freedom ($s_1 = 0.65, s_2 = 1.32$)	1/38	1/34	37
(4) Fused lasso, 38 degrees of freedom ($s_1 = 1.08, s_2 = 0.71$)	1/38	2/34	135
(5) Fused lasso, 20 degrees of freedom ($s_1 = 1.35, s_2 = 1.01$)	1/38	4/34	737
(6) Fusion, 1 degree of freedom	1/38	12/34	975

In the case where we are given a data set with unordered features, we would have to estimate the order of the features to use the fused lasso since it assumes ordering. The article suggests estimating the orders for the features using multidimensional scaling or hierarchical clustering. This method of estimation was tested to classify leukemia by using microarrays. The training dataset consists of 7129 genes and 38 samples with 27 in class 1 and 11 in class 2 and the test dataset consist of 34 samples. The results of the analysis are in the table above. We see that the fussed lasso seems to have a lower test error compared the fusion model, but lasso seems to have a lower test error compared to all models. This implies that fused lasso will not necessarily have better prediction compared to lasso and fusion model when the features don't have a meaningful order.

5.2 Runtime



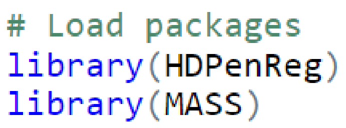
p	N	*Start*	*Time (s)*
100	20	Cold	0.09
500	20	Cold	1.0
1000	20	Cold	2.0
1000	200	Cold	30.4
2000	200	Cold	120.0
2000	200	Warm	16.6

Although the fused lasso is powerful in terms of predicting responses compared to lasso in certain cases, the runtime of the algorithm could become a practical limitation especially if you use five or ten fold cross validation. In the article, a test was conducted to monitor the runtime of the fused model. The results of this test are in the above table which show computation times for datasets of various dimensions, on a 2.4 GHz Xeon Linux computer. We see when the p is doubles to 2000 on a cold start, the time to run the fused lasso model seem to quadruple the time. Although a hot start can help with large values for the bound but this is not always practical.

6. The Simulation Study

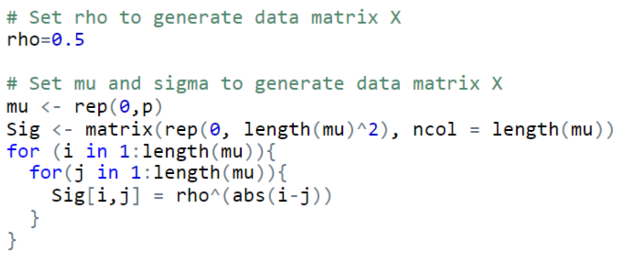
We carried out a simulation study very similar to the one conducted in the article to compare the lasso and fused lasso performance.

6.1 Setup



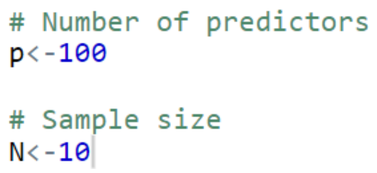
```
# Load packages
library(HDPenReg)
library(MASS)
```

For the simulation, we used the MASS and hdpenreg libraires in R for which we have attached a reference to the documentation below.



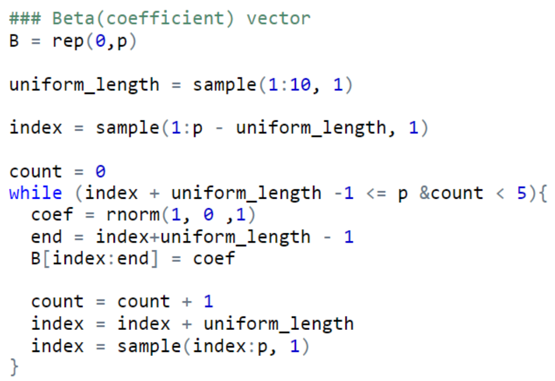
```
# Set rho to generate data matrix X
rho=0.5

# Set mu and sigma to generate data matrix X
mu <- rep(0,p)
Sig <- matrix(rep(0, length(mu)^2), ncol = length(mu))
for (i in 1:length(mu)){
  for(j in 1:length(mu)){
    Sig[i,j] = rho^(abs(i-j))
  }
}
```

Since we did not have access to prostate data set used in the simulation in the article and we did not have any information about the mean and covariances of the data, we defined our own mean and covariance matrix to generate the data matrix X which has $N\ x\ p$ dimension. The mean vector μ was defined as vector of size p with all zeros and the covariance matrix σ of dimensions $p\ x\ p$ is defined where $\sigma_{ij} = rho^{|i-j|}$ where rho is defined as 0.5, like how it is defined in tutorial 7.



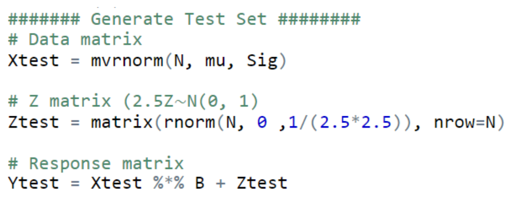
```
# Number of predictors
p<-100

# Sample size
N<-10
```

We set the dimension for the data matrix to be $N = 10$ and $p = 100$ which is different from the article so we could run the simulation in reasonable amount of time but still lets us test the performance of the model when $p \gg N$.



```
### Beta(coefficient) vector
B = rep(0,p)

uniform_length = sample(1:10, 1)

index = sample(1:p - uniform_length, 1)

count = 0
while (index + uniform_length -1 <= p &count < 5){
  coef = rnorm(1, 0 ,1)
  end = index+uniform_length - 1
  B[index:end] = coef

  count = count + 1
  index = index + uniform_length
  index = sample(index:p, 1)
}
```

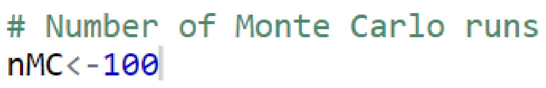
We then generated coefficient vectors β like described in article with a similar structure but scaled down since we are using a small data matrix. The coefficient vector is defined by choosing 1–5 non-overlapping m/z sites at random and defining blocks of equal non-zero coefficients of lengths uniform between 1 and 10. The values of the coefficients were generated as $N(0,1)$.



```
####### Generate Test Set ########
# Data matrix
Xtest = mvrnorm(N, mu, Sig)

# Z matrix (2.5Z~N(0, 1)
Ztest = matrix(rnorm(N, 0 ,1/(2.5*2.5)), nrow=N)

# Response matrix
Ytest = Xtest %*% B + Ztest
```

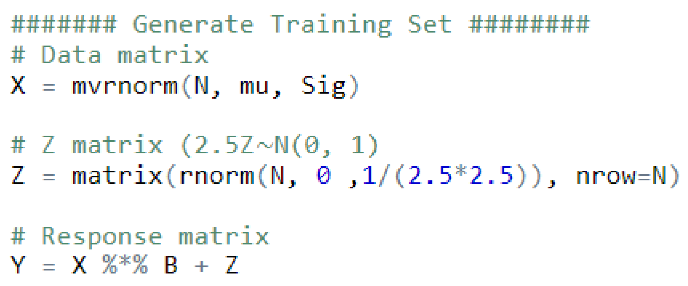
We also defined the test set before the simulation, like the training set describes later.

6.2 Monte Carlo Runs



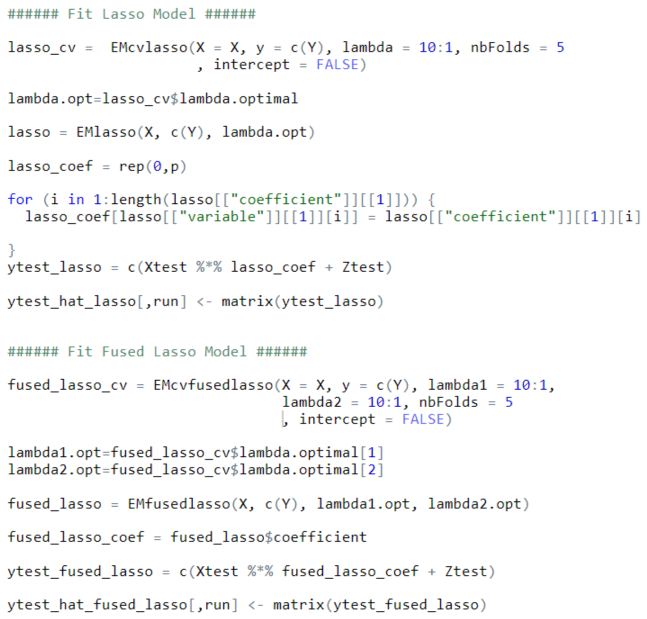
```
# Number of Monte Carlo runs
nMC<-100
```

To conduct the simulation, we ran 100 runs of the Monte Carlo simulation to estimate the prediction for the test data.



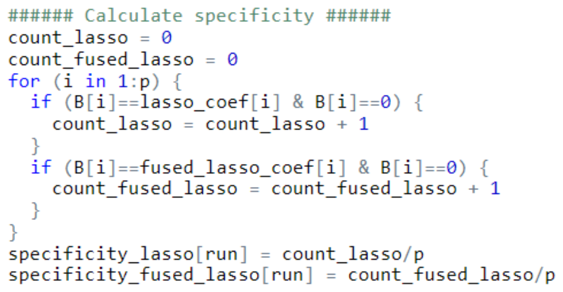
```
####### Generate Training Set ########
# Data matrix
X = mvrnorm(N, mu, Sig)

# Z matrix (2.5Z~N(0, 1)
Z = matrix(rnorm(N, 0 ,1/(2.5*2.5)), nrow=N)

# Response matrix
Y = X %*% B + Z
```

For each run, we define a new training set, where the response is defined as $Y = X\beta + Z$ with X and β generated as described above and $2.5Z{\sim}N(0,1)$.



```
###### Fit Lasso Model ######

lasso_cv =  EMcvlasso(X = X, y = c(Y), lambda = 10:1, nbFolds = 5
                      , intercept = FALSE)

lambda.opt=lasso_cv$lambda.optimal

lasso = EMlasso(X, c(Y), lambda.opt)

lasso_coef = rep(0,p)

for (i in 1:length(lasso[["coefficient"]][[1]])) {
  lasso_coef[lasso[["variable"]][[1]][i]] = lasso[["coefficient"]][[1]][i]

}
ytest_lasso = c(Xtest %*% lasso_coef + Ztest)

ytest_hat_lasso[,run] <- matrix(ytest_lasso)


###### Fit Fused Lasso Model ######

fused_lasso_cv = EMcvfusedlasso(X = X, y = c(Y), lambda1 = 10:1,
                                lambda2 = 10:1, nbFolds = 5
                                , intercept = FALSE)

lambda1.opt=fused_lasso_cv$lambda.optimal[1]
lambda2.opt=fused_lasso_cv$lambda.optimal[2]

fused_lasso = EMfusedlasso(X, c(Y), lambda1.opt, lambda2.opt)

fused_lasso_coef = fused_lasso$coefficient

ytest_fused_lasso = c(Xtest %*% fused_lasso_coef + Ztest)

ytest_hat_fused_lasso[,run] <- matrix(ytest_fused_lasso)
```

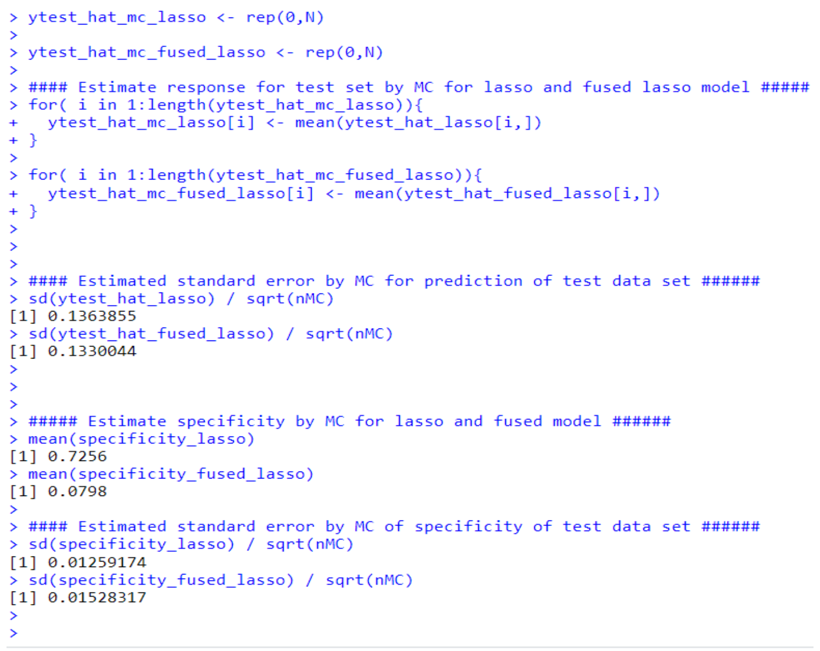
After the training data is generated, for each run we find the optimal coefficients using the lasso and fused model and then use it to make predictions for the test data set and store it in a matrix. For the lasso and fused model, we try to use a similar search strategy as described earlier by set the bounds to be restricted in the range of 1-10 and then using 5 folds cross validation to find the optimal bounds for both models. We used the expectation-maximization (EM) algorithm instead of the LARS algorithm to compute the lasso and fused model since there was not an ideal package which implemented the LARS algorithm for the fused lasso but still gives similar conclusions. The functions for the EM algorithm for lasso and fused lasso are defined in the hdpenreg library and tries to maximize the likelihood of the data.



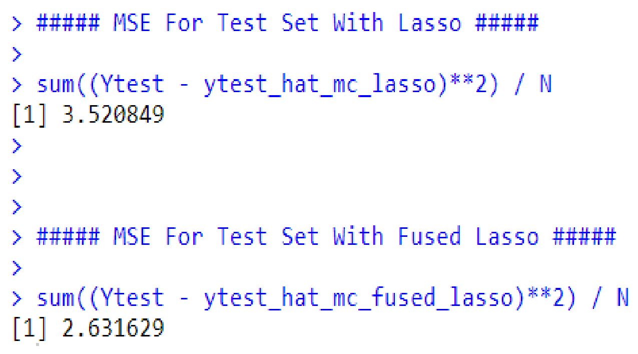
```
###### Calculate specificity ######
count_lasso = 0
count_fused_lasso = 0
for (i in 1:p) {
  if (B[i]==lasso_coef[i] & B[i]==0) {
    count_lasso = count_lasso + 1
  }
  if (B[i]==fused_lasso_coef[i] & B[i]==0) {
    count_fused_lasso = count_fused_lasso + 1
  }
}
specificity_lasso[run] = count_lasso/p
specificity_fused_lasso[run] = count_fused_lasso/p
```

After finding the models, for each run we calculated the specificity which is the proportion of true zero coefficients that are detected by each method and stored it in a matrix.

Finally, to conclude the simulation after the 100 Monte Carlo simulation, we calculate the average prediction, the average specificity, the estimated standard error, and the mean squared error of the mean predictions for both models. The output of the R code is shown in the images below



```
> ytest_hat_mc_lasso <- rep(0,N)
>
> ytest_hat_mc_fused_lasso <- rep(0,N)
>
> #### Estimate response for test set by MC for lasso and fused lasso model #####
> for( i in 1:length(ytest_hat_mc_lasso)){
+   ytest_hat_mc_lasso[i] <- mean(ytest_hat_lasso[i,])
+ }
>
> for( i in 1:length(ytest_hat_mc_fused_lasso)){
+   ytest_hat_mc_fused_lasso[i] <- mean(ytest_hat_fused_lasso[i,])
+ }
>
>
>
> #### Estimated standard error by MC for prediction of test data set ######
> sd(ytest_hat_lasso) / sqrt(nMC)
[1] 0.1363855
> sd(ytest_hat_fused_lasso) / sqrt(nMC)
[1] 0.1330044
>
>
>
> ##### Estimate specificity by MC for lasso and fused model ######
> mean(specificity_lasso)
[1] 0.7256
> mean(specificity_fused_lasso)
[1] 0.0798
>
> #### Estimated standard error by MC of specificity of test data set ######
> sd(specificity_lasso) / sqrt(nMC)
[1] 0.01259174
> sd(specificity_fused_lasso) / sqrt(nMC)
[1] 0.01528317
>
>
```



```
> ##### MSE For Test Set With Lasso #####
>
> sum((Ytest - ytest_hat_mc_lasso)**2) / N
[1] 3.520849
>
>
>
> ##### MSE For Test Set With Fused Lasso #####
>
> sum((Ytest - ytest_hat_mc_fused_lasso)**2) / N
[1] 2.631629
```

6.3 Results

The results of the simulation are summarized in the table below.

Method	***Test Error***	***Specificity***
Lasso	*3.520849(0.1363855)*	*0.7256(0.01259174)*
Fused Lasso	*2.631629(0.1330044)*	*0.0798(0.01528317)*

Standard errors estimates are given in parentheses.

6.4 Conclusion

We see that the fused lasso improves on the test error by roughly 0.9 and has slightly smaller standard error compared to lasso which is like the conclusion from the article. On the other hand, lasso model seems to detect a much higher proportion of true zero coefficients with similar standard error for specificity compares to fused lasso. Overall, the fused lasso seems to perform well under the case where $p \gg N$.

7. References

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J. and Knight, K. (2005), *Sparsity and smoothness via the fused lasso*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67: 91-108. https://doi.org/10.1111/j.1467-9868.2005.00490.x

The Comprehensive R Archive Network. (n.d.). Retrieved April 22, 2022, from https://cran.r-project.org/web/packages/HDPenReg/HDPenReg.pdf