Challenge: Multi-Stage Lasso and Post-Selection Inference

Lingtian Bu*
WISE 2017

Update: December 8, 2019

Abstract

In this assignment, I first introduce the original model setting. Second, simulation with R to compare the performance of the Lasso (alone) and the relaxed Lasso (including post-Lasso OLS) are presented. Then, I change the setting and do the regressions again. Finally, comparing the results of different settings helps me update the conclusion.

Keywords: Lasso multi-stage Lasso high-dimensional sparsity

1 Multi-Stage Lasso

Consider the following linear model:

$$y = x'\beta + e \tag{1}$$

Let $\hat{\beta}$ be the population least squares coefficients:

$$\hat{\beta} = \arg\min_{\beta} E\left[(y - x'\beta)^2\right] \tag{2}$$

When *beta* is high-dimensional but contains mostly zeros, i.e. when the number of non-zero population coefficients $\ll p$ (dimension of *beta*), we say there is sparsity.

In high dimensional sparse settings, there's possibility to improve upon the results of lasso by running a multi-stage lasso, e.g. the two-stage Lasso- **Relaxed Lasso**:

Alogorithm. *Relaxed Lasso(including Post-Lasso OLS)*

Stage 1 *In the first stage, estimate equation (1) by th Lasso*

Stage 2 *Run Lasso (or OLS) again on the variables selected by the first stage. (When the optimal penalty is close to 0, then this stage becomes OLS)*

Intuitively, the first stage focus on variable selection and the second stage focus on estimation.

* 卜令天 15220172202239

2 The Original Setting

Suppose our "true" model satisfies the following conditions:

$$y = \beta_1 x_1 + \cdots + \beta_{139} x_{139} + e \tag{3}$$

And our independent varaibles as well as the error term follows i.i.d. standard normal distribution, i.e.

$$x_1, \ldots, x_{139}, e \sim^{i.i.d.} \mathrm{N}(0, 1)$$

Also, our true underlining population function has the true parameters:

$$\beta_1 = \cdots = \beta_7 = 3$$

$$\beta_8 = \cdots = \beta_{139} = 0$$

```
rm(list=ls())
set.seed(123)

# Training Set
n = 140 # sample size
p = 139 # number of variables
s = 7 # number of variables witn non-zero coefficients
X = matrix(rnorm(n*p),ncol=p)
beta = c(rep(3,s),rep(0,p-s))
Y = X%*%beta + rnorm(n)
```

```
# Test set
nn = 1e5 # testset size
Xnew = matrix(rnorm(nn*p), ncol=p)
Ynew = Xnew%*%beta + rnorm(nn)
```

2.1 OLS

First, I run the simple OLS to see whether in high-dimensional sparse settings OLS still works or not.

```
#######
# OLS #
#######
fit.ols = lm(Y ~ X-1)

# number of non-zero coefs
sum(fit.ols$coefficients!=0)

##[1] 139
```

```
library(lmtest)
coeftest(fit.ols)[1:10,]

##      Estimate   Std. Error t value  Pr(>|t|)
##X1    1.6494053  1.7478108  0.9436979 0.5184355
##X2    3.9969040  1.6064242  2.4880751 0.2432893
##X3    1.9536392  1.4901515  1.3110339 0.4148314
##X4    2.5319031  0.7067733  3.5823412 0.1732991
##X5    2.3475021  1.1844137  1.9819951 0.2974763
##X6    3.3111564  0.7084816  4.6735957 0.1341928
##X7    2.3097514  1.0728085  2.1529950 0.2768158
##X8   -0.9024125  0.8359239 -1.0795390 0.4756622
##X9    0.1619608  0.6030577  0.2685660 0.8329670
##X10 -0.1550563  0.6607082 -0.2346820 0.8532524

# number of significant vars
pv.ols = summary(fit.ols)$coefficients[,4]
sum(pv.ols<.05)

##[1] 0

# test err
pred.ols = predict(fit.ols,data.frame(X=Xnew))
err.ols = mean((Ynew - pred.ols)^2)
err.ols

##[1] 123.2682
```

As we can see, the OLS predicts that there isn't significant coefficients which is not consistent with our setting. Even worse, there's large error for test data. Though our "true" model is drawn from linear model, in high-dimensional sparse setting just run OLS will show nothing.

2.2 Lasso

Second I run Lasso alone. The first step is to select the optimal λ by using cross-validation method. The change of in-sample error as λ increasing is shown in Figure 1

```
#########
# LASSO #
#########

require(glmnet)
cv.lasso = cv.glmnet(X,Y,intercept=FALSE,alpha=1)
```

```
lambda.star = cv.lasso$lambda.1se # Alternatively: cv.lasso$lambda.min
lambda.star

## [1] 0.1940468

plot(cv.lasso)
```

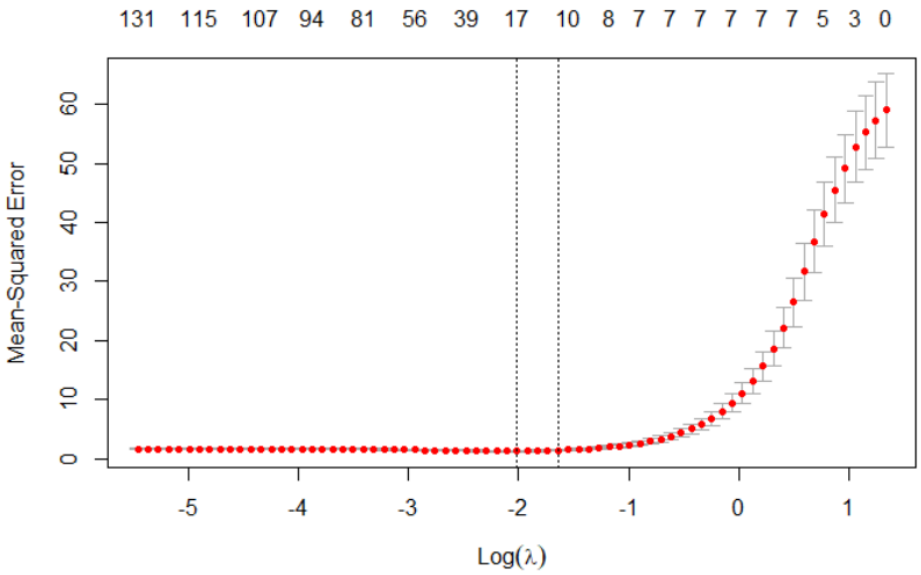

Figure 1: Pick the optimal λ

Also, I plot the figure which shows how the coefficients change with λ, see Figure 2.

```
lasso.all = glmnet(X,Y,intercept=FALSE,alpha=1)
plot(lasso.all,xvar = "lambda")
```

```
fit.lasso = glmnet(X,Y,intercept=FALSE,alpha=1,lambda=lambda.star)
sum(fit.lasso$beta!=0)

##[1] 11

pred.lasso = predict(fit.lasso,Xnew)
err.lasso = mean((Ynew - pred.lasso)^2)
err.lasso

##[1] 1.237999
```

By running Lasso, I found that the number of significant coefficients is larger than true model. However, comparing OLS the test error decrease so much.

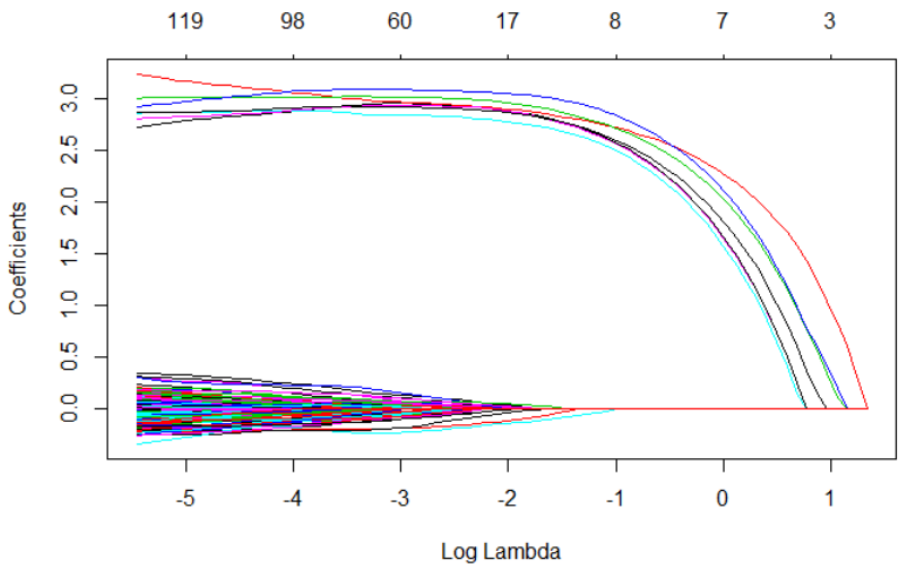

Figure 2: Change of coefficients when λ going up

2.3 Post-Lasso OLS

In applied economics literature, the post-lasso OLS has gained some popularity. On the one hand, it provides statistical inference for the coefficients of interest. On the other hand, such inference is generally invalid since the second stage model is selected by the first stage. The problem of assessing the strength of the evidence after searching through a large number of models to find the best one is called post-selection inference or selective inference.

In this section, I run the post-Lasso OLS and do the post-selection inference as well as compare the naive P-value and adjusted p-value.

```
########################################
# Post-selection inference for the Lasso#
########################################

# Selective Inference
require(selectiveInference)
lassoInf = fixedLassoInf(X,Y,intercept = FALSE,
beta=fit.lasso$beta,
lambda=fit.lasso$lambda)
pv.lasso = lassoInf$pv
pv.lasso

##[1] 2.513465e-05 2.790539e-06 4.783584e-06 1.001095e-05
##[5] 7.839593e-05 1.034882e-02 8.896250e-04 6.831787e-01
##[9] 7.491452e-01 8.977512e-01 8.053963e-01
```

```
#Compare with the naive method of running OLS on variables selected by the Lasso
# Post-lasso OLS
chosen.var = X[,which(fit.lasso$beta!= 0)]
fit.olslasso = lm(Y ~ chosen.var-1)
pv.naive = summary(fit.olslasso)$coefficients[,4]
pv.naive

##chosen.var1  chosen.var2  chosen.var3  chosen.var4
##1.345676e-72 1.401549e-73 3.816866e-77 1.769233e-70

##chosen.var5  chosen.var6  chosen.var7  chosen.var8
##7.786055e-65 2.311502e-67 1.468382e-69 1.059704e-02

##chosen.var9 chosen.var10 chosen.var11
##8.077388e-03 3.891166e-01 1.859411e-01
```

```
pred.olslasso = predict(fit.olslasso,data.frame(X=Xnew))
err.olslasso = mean((Ynew - pred.olslasso)^2)
err.olslasso

##[1] 122.5348
```

Apparently, naive p-values are smaller than adjusted p-value.

```
#Post-selection Inference(Adjusted)

lassoInf

## Call:
## fixedLassoInf(x = X, y = Y, beta = fit.lasso$beta, lambda = fit.lasso$lambda,
## intercept = FALSE)

## Standard deviation of noise (specified or estimated) sigma = 7.728

## Testing results at lambda = 0.194, with alpha = 0.100

## Var   Coef Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
## 1    3.048   4.598   0.000     1.941    4.701       0.049      0.049
## 2    2.995   4.686   0.000     1.937    4.049       0.049      0.050
## 3    3.112   5.018   0.000     2.069    4.334       0.048      0.048
## 4    3.162   4.412   0.000     2.040    6.135       0.049      0.050
## 5    2.944   3.947   0.000     1.743    5.350       0.050      0.050
## 6    3.062   4.150   0.010     1.122    6.797       0.050      0.050
## 7    3.029   4.333   0.001     1.804    7.595       0.049      0.050
```

```
## 11   -0.218   -0.310    0.683     -2.406     6.658        0.050       0.050
## 26   -0.235   -0.321    0.749     -1.157     6.683        0.050       0.050
## 100   0.069    0.103    0.898    -19.504     0.827        0.050       0.050
## 109   0.121    0.159    0.805    -14.353     2.305        0.050       0.050

Note: coefficients shown are partial regression coefficients

#Directly run OLS(Naive)

coeftest(fit.olslasso)

## t test of coefficients:

##              Estimate Std. Error t value  Pr(>|t|)
## chosen.var1   3.048209   0.079165 38.5043 < 2.2e-16 ***
## chosen.var2   2.994728   0.076311 39.2436 < 2.2e-16 ***
## chosen.var3   3.112095   0.074046 42.0290 < 2.2e-16 ***
## chosen.var4   3.161964   0.085576 36.9493 < 2.2e-16 ***
## chosen.var5   2.943976   0.089056 33.0574 < 2.2e-16 ***
## chosen.var6   3.061809   0.088092 34.7571 < 2.2e-16 ***
## chosen.var7   3.028710   0.083456 36.2913 < 2.2e-16 ***
## chosen.var8  -0.217985   0.084050 -2.5935  0.010597 *
## chosen.var9  -0.235134   0.087392 -2.6906  0.008077 **
## chosen.var10  0.069086   0.079948  0.8641  0.389117
## chosen.var11  0.120799   0.090842  1.3298  0.185941
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Coefficient inference results are the same for two kinds of methods.

2.4 Relaxed Lasso

Finally I run the relaxed Lasso.

```
#################
# Relaxed Lasso #
#################
require(relaxo)
fit.relaxo = cvrelaxo(X,Y)
sum(fit.relaxo$beta!=0)

##[1] 8

pred.relaxo = predict(fit.relaxo,Xnew)
err.relaxo = mean((Ynew - pred.relaxo)^2)
```

```
err.relaxo

##[1] 1.117397
```

2.5 Summary

The test error for different methods are summarized in Table 1. It's obvious that Lasso decrease the test error in a large extense though it has some bias.

Method	OLS	Lasso	Post-Lasso OLS	Relaxed Lasso
Error	123.27	1.24	122.53	1.12

Table 1: Summary for test error

3 Change the Size and Sparsity of β

Suppose our "true" model satisfies the following conditions:

$$y = \beta_1 x_1 + \cdots + \beta_{49} x_{49} + e \tag{4}$$

And our independent varaibles as well as the error term follows i.i.d. standard normal distribution, i.e.

$$x_1, \ldots, x_{49}, e \sim^{i.i.d.} \mathrm{N}(0, 1)$$

Also, our true underlining population function has the true parameters:

$$\beta_1 = \cdots = \beta_7 = 3$$

$$\beta_8 = \cdots = \beta_{49} = 0$$

```
# Training Set
n = 50 # sample size
p = 49 # number of variables
s = 7 # number of variables witn non-zero coefficients
X = matrix(rnorm(n*p),ncol=p)
beta = c(rep(3,s),rep(0,p-s))
Y = X%*%beta + rnorm(n)

# Test set
nn = 1e5 # testset size
Xnew = matrix(rnorm(nn*p), ncol=p)
Ynew = Xnew%*%beta + rnorm(nn)
```

3.1 OLS

```
#######
# OLS #
#######
fit.ols = lm(Y ~ X-1)

# number of non-zero coefs
sum(fit.ols$coefficients!=0)

##[1] 49

library(lmtest)
coeftest(fit.ols)[1:10,]

## Estimate Std. Error   t value   Pr(>|t|)
## X1   2.7699390 0.07903916 35.045149 0.01816078
## X2   1.6963592 0.05319080 31.891968 0.01995522
## X3   3.5099793 0.12327945 28.471730 0.02235053
## X4   2.9498911 0.05846650 50.454379 0.01261608
## X5   3.2036968 0.09390654 34.115800 0.01865521
## X6   3.1195216 0.10676579 29.218363 0.02177984
## X7   2.8074958 0.06087360 46.120085 0.01380136
## X8  -0.5641914 0.06201159 -9.098161 0.06969261
## X9   0.1650855 0.07791348  2.118831 0.28072618
## X10 -0.4748158 0.16357157 -2.902802 0.21120624

pv.ols = summary(fit.ols)$coefficients[,4]
sum(pv.ols<.05)

##[1] 8

pred.ols = predict(fit.ols,data.frame(X=Xnew))
err.ols = mean((Ynew - pred.ols)^2)
err.ols

##[1] 136.3339
```

This time the sparsity is smaller than before. As we can see, the OLS predicts that there is 8 significant coefficients. Though our "true" model is set 7 significant coefficients and there's some bias, linear model somehow works.

3.2 Lasso

```
#########
# LASSO #
#########

require(glmnet)
cv.lasso = cv.glmnet(X,Y,intercept=FALSE,alpha=1)
lambda.star = cv.lasso$lambda.1se # Alternatively: cv.lasso$lambda.min
lambda.star

##[1] 0.2404726
```

```
plot(cv.lasso)
```

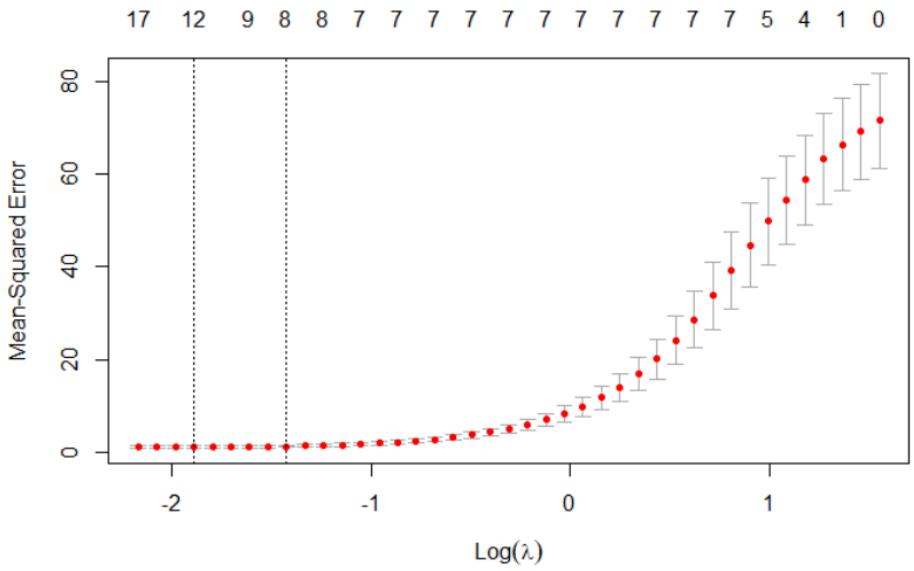

Figure 3: Pick the optimal λ

```
fit.lasso = glmnet(X,Y,intercept=FALSE,alpha=1,lambda=lambda.star)
lasso.all = glmnet(X,Y,intercept=FALSE,alpha=1)
plot(lasso.all,xvar = "lambda")
```

```
sum(fit.lasso$beta!=0)

##[1] 8

pred.lasso = predict(fit.lasso,Xnew)
err.lasso = mean((Ynew - pred.lasso)^2)
err.lasso

##[1] 1.883291
```

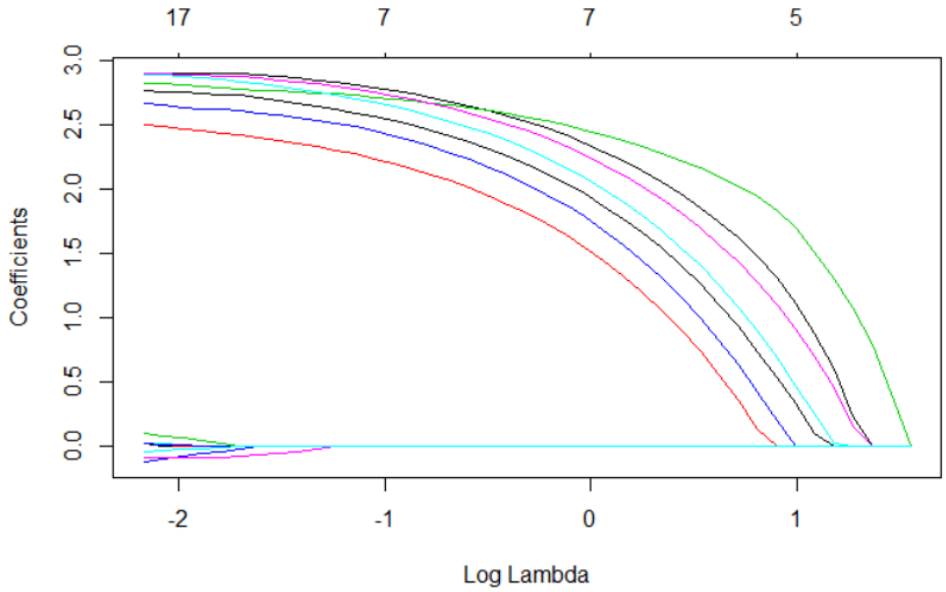

Figure 4: Change of coefficients when λ going up

3.3 Post-Lasso OLS

```
##########################################
# Post-selection inference for the Lasso#
##########################################
# Selective Inference
require(selectiveInference)
lassoInf = fixedLassoInf(X,Y,intercept = FALSE,
beta=fit.lasso$beta,
lambda=fit.lasso$lambda)
pv.lasso = lassoInf$pv
pv.lasso

##[1] 0.033493562 0.040537027 0.025891398 0.036187205
##[5] 0.052934796 0.008397049 0.016385302 0.878732582


#Compare with the naive method of running OLS on variables selected by the Lasso
# Post-lasso OLS
chosen.var = X[,which(fit.lasso$beta!= 0)]
fit.olslasso = lm(Y ~ chosen.var-1)
pv.naive = summary(fit.olslasso)$coefficients[,4]
pv.naive

##chosen.var1  chosen.var2  chosen.var3  chosen.var4
##3.336159e-26 1.904578e-25 5.398197e-27 4.325312e-25
```

```
##chosen.var5  chosen.var6  chosen.var7  chosen.var8
##5.127744e-27 6.611227e-30 2.021765e-28 8.611812e-02
```

```
lassoInf

## Call:
## fixedLassoInf(x = X, y = Y, beta = fit.lasso$beta, lambda = fit.lasso$lambda,
## intercept = FALSE)

## Standard deviation of noise (specified or estimated) sigma = 8.548

## Testing results at lambda = 0.240, with alpha = 0.100

## Var   Coef  Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
## 1    3.029    2.127   0.033     0.334    5.391       0.049      0.049
## 2    2.600    2.034   0.041     0.169    6.297       0.050      0.049
## 3    2.843    2.228   0.026     0.482    5.377       0.050      0.049
## 4    2.767    1.992   0.036     0.298   10.333       0.049      0.050
## 5    3.022    2.231   0.053    -0.069    5.267       0.050      0.048
## 6    3.011    2.636   0.008     1.014    5.255       0.049      0.049
## 7    2.912    2.420   0.016     0.724    4.899       0.049      0.049
## 27 -0.223   -0.155   0.879    -1.391   28.061       0.049      0.050

## Note: coefficients shown are partial regression coefficients

coeftest(fit.olslasso)

## t test of coefficients:

##              Estimate Std. Error t value Pr(>|t|)
## chosen.var1   3.02937    0.12581 24.0798  < 2e-16 ***
## chosen.var2   2.59956    0.11287 23.0307  < 2e-16 ***
## chosen.var3   2.84266    0.11271 25.2207  < 2e-16 ***
## chosen.var4   2.76709    0.12271 22.5506  < 2e-16 ***
## chosen.var5   3.02241    0.11968 25.2536  < 2e-16 ***
## chosen.var6   3.01053    0.10087 29.8457  < 2e-16 ***
## chosen.var7   2.91211    0.10628 27.3993  < 2e-16 ***
## chosen.var8  -0.22272    0.12672 -1.7575  0.08612 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pred.olslasso = predict(fit.olslasso,data.frame(X=Xnew))
err.olslasso = mean((Ynew - pred.olslasso)^2)
```

```
err.olslasso

##[1] 135.8986
```

3.4 Relaxed Lasso

```
##################
# Relaxed Lasso #
##################
require(relaxo)
fit.relaxo = cvrelaxo(X,Y)
sum(fit.relaxo$beta!=0)

##[1] 8

pred.relaxo = predict(fit.relaxo,Xnew)
err.relaxo = mean((Ynew - pred.relaxo)^2)
err.relaxo

##[1] 1.303492
```

3.5 Summary

When sparsity is smaller, the test error results are summarized in Table 2

Method	OLS	Lasso	Post-Lasso OLS	Relaxed Lasso
Error	136.33	1.88	135.90	1.30

Table 2: Summary for test error

All test errors are larger than before.

4 References

[1] Model Selection and Regularization. Jiaming Mao

[2] Belloni, A. and V. Chernozhukov. 2013. "Least squares after model selection in high dimensional sparse models," Bernoulli, 19(2).

[3] Lee, J. D., D. L. Sun, Y. Sun, and J. E. Taylor. 2016. “Exact Post-selection Inference with Application to the Lasso,” The Annals of Statistics, 44(3).

[4] Meinshausen, N. 2007. “Relaxed Lasso,” Computational Statistics and Data Analysis, 52(1).