Selective Inference for LASSO with Overconditioning

Nguyen Quoc Khanh
24520793
University of Information Technology,
Ho Chi Minh City, Vietnam

14/05/2025

Abstract

Exact post-selection inference for the LASSO with Selective Inference technique. The basic idea behind it is to make conditional inference on the selection event. This note will only used Lee et al. 2016 for reference, the notation may vary.

1 Theories

1.1 Setting

To formally formulate the problem, we consider the random response vector

$$\boldsymbol{Y} = (Y_1, ..., Y_n)^\top \sim N(\mu, \Sigma) \tag{1}$$

1.2 Feature selection and its selection event

The LASSO optimization problem is given:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2N} \left\| y^{\text{obs}} - X\beta \right\|_2^2 + \lambda \left\| \beta \right\|_1 \right\} \tag{2}$$

Since LASSO produces a sparse solution, we characterize the active set as:

$$\mathcal{A}^{\text{obs}} = \mathcal{A}\big(y^{\text{obs}}\big) = \big\{ j : \beta_j \neq 0 \big\} \tag{3}$$

The event that the active set is the same with observed active set is written as

$$\mathcal{A}(\boldsymbol{Y}) = \mathcal{A}\big(y^{\text{obs}}\big) \tag{4}$$

1.3 Formulate Hypothesis and Test Statistics

Test statistics:

The j-th entry of β is can mathematically derive as follow

$$\hat{\beta}_j = X_{\mathcal{A}^{\text{obs}}} \big(X_{\mathcal{A}^{\text{obs}}}^\top X_{\mathcal{A}^{\text{obs}}} \big)^{-1} e_j y^{\text{obs}} = \eta_j^\top y^{\text{obs}} \tag{5}$$

where e_j is a vector where j-th entry is 1 and else are 0

Hypothesis:

$$\mathbb{H}_{0,j} : \eta_j^\top \mu = 0 \quad , \quad \mathbb{H}_{1,j} : \eta_j^\top \mu \neq 0 \tag{6}$$

Since the hypothesis is generated by data, selection bias will exist and we need to address this with Selective Inference method to produce a valid inference.

1.4 Statistical Inference

Distribution of the Test Statistics

$$\eta_j^\top \mu \mid \{A(\boldsymbol{Y}) = \mathcal{A}^{\text{obs}} \ , \ \ \mathcal{Q}(\boldsymbol{Y}) = \mathcal{Q}\big(y^{\text{obs}}\big)\} \tag{7}$$

where $\mathcal{Q}(\boldsymbol{Y}) = \Big(I_n - c\eta_j^\top\Big)\boldsymbol{Y}$ with $c = \Sigma\eta_j\Big(\eta_j^\top\Sigma\eta_j\Big)^{-1}$.

The second condition indicates the components that is independent of the test statistic is the same in both response vectors.

Characterize the Selection Event

KKT conditions of the Lasso:

$$\begin{aligned} &X^T\Big(X\hat{\beta} - Y\Big) + \lambda S = 0, \\ &S_j = \text{sign}\Big(\hat{\beta}_j\Big), \text{ if } \hat{\beta}_j \neq 0, \\ &S_j \in (-1, 1), \quad \text{ if } \hat{\beta}_j = 0, \end{aligned} \tag{8}$$

We can rewrite the (8) by partioning them according to set A^{obs}, and adopting A_c^{obs} means "variables not in A_c^{obs}":

$$\begin{aligned} X_{A^{\text{obs}}}^\top\Big(X_{A^{\text{obs}}}\hat{\beta}_{A^{\text{obs}}} - Y\Big) + \lambda S_{A^{\text{obs}}} = 0, \\ X_{A_c^{\text{obs}}}^\top\Big(X_{A_c^{\text{obs}}}\hat{\beta}_{A_c^{\text{obs}}} - Y\Big) + \lambda S_{A_c^{\text{obs}}} = 0, \\ S_{A^{\text{obs}}} = \text{sign}\Big(\hat{\beta}_{A^{\text{obs}}}\Big), \\ \|S_{A_c^{\text{obs}}}\|_\infty < 1 \end{aligned} \tag{9}$$

Since the KKT conditions are necessary and sufficient for a solution, we obtain that $\{A = A^{\text{obs}}, S = S^{\text{obs}}\}$ (for the sake of simplicity, lets $A = A(Y)$) if and only if

$$\begin{aligned} X_A^\top\Big(X_A\hat{\beta}_A - Y\Big) + \lambda S_A = 0, \\ X_{A_c}^\top\Big(X_{A_c}\hat{\beta}_{A_c} - Y\Big) + \lambda S_{A_c} = 0, \\ S_A = \text{sign}\Big(\hat{\beta}_A\Big), \\ \|S_{A_c}\|_\infty < 1 \end{aligned} \tag{10}$$

By solving (10) for $\hat{\beta}_A$ and S_{A_c}, we obtain the equivalent set of conditions:

$$\begin{aligned} \hat{\beta}_A = \big(X_A^\top X_A\big)^{-1}\big(X_A^\top Y - \lambda S_A\big), \\ S_{A_c} = X_{A_c}^\top\Big(X_{A_c}^\top\Big)^+ S_A + \frac{1}{\lambda}X_{A_c}^\top\Big(I_n - X_{A_c}\Big(X_{A_c}\Big)^+\Big)Y, \\ \text{sign}\Big(\hat{\beta}_A\Big) = S_A, \\ \|S_{A_c}\|_\infty < 1, \end{aligned} \tag{11}$$

where $X^+ = \big(X^\top X\big)^{-1}X^\top, \big(X^\top\big)^+ = X\big(X^\top X\big)^{-1}$. We then have the following selection event:

$$\{A = A^{\text{obs}}, S = S^{\text{obs}}\} = \{\mathcal{A}(A, S)y \leq b(A, S)\} \tag{12}$$

with:

$$\mathcal{A}(A,S) = \begin{pmatrix} \frac{1}{\lambda}X_{A_c}^\top\Big(I - X_A(X_A)^+\Big) \\ -\frac{1}{\lambda}X_{A_c}^\top\Big(I - X_A(X_A)^+\Big) \\ -\text{diag}(S)\big(X_A^\top X_A\big)^{-1}X_A^\top \end{pmatrix} \ , \ b(A,S) = \begin{pmatrix} 1 - X_{A_c}^\top\big(X_A^\top\big)^+ S \\ 1 + X_{A_c}^\top\big(X_A^\top\big)^+ S \\ -\lambda\ \text{diag}(S)\big(X_A^\top X_A\big)^{-1}S \end{pmatrix} \tag{13}$$

1.5 Polyhedra conditioning sets

The (over)conditioning event on a single polyhedron $\{A = A^{\text{obs}}, S = S^{\text{obs}}\}$, by conditioning on both the model and signs. To understand the distribution of

$$\eta^\top Y \mid \{\mathcal{A}Y \leq b\} \tag{14}$$

we rewrite $\{\mathcal{A}Y \leq b\}$ in terrms of $\eta^\top y$ and a component z which is independent of $\eta^\top y$. That component is

$$z = (I - c\eta^\top)Y, \text{ with } c = \Sigma\eta(\eta^\top\Sigma\eta)^{-1} \tag{15}$$

Then the conditioning set be rewritten as follows:

$$\{\mathcal{A}Y \leq b\} = \{\mathcal{V}^-(z) \leq \eta^\top Y \leq \mathcal{V}^+(z), \mathcal{V}^0(z) \geq 0\} \tag{16}$$

where

$$\mathcal{V}^-(z) = \max_{j:(\mathcal{A}c)_j<0} \frac{b_j - (\mathcal{A}z)_j}{(\mathcal{A}c)_j}, \tag{17}$$

$$\mathcal{V}^+(z) = \min_{j:(\mathcal{A}c)_j>0} \frac{b_j - (\mathcal{A}z)_j}{(\mathcal{A}c)_j}, \tag{18}$$

$$\mathcal{V}^0(z) = \min_{j:(\mathcal{A}c)_j=0} b_j - (\mathcal{A}z)_j \tag{19}$$

Then we can have the distribution of the test statistics is follow the Truncated Normal:

$$[\eta^\top y \mid \mathcal{A}Y \leq b, z = z_0] \sim \text{TN}(\eta^T\mu, \sigma^2 \|\eta\|^2, \mathcal{V}^-(z_0), \mathcal{V}^+(z_0)) \tag{20}$$

2 The funny coding part

2.1 Generate data

```python
import numpy as np
def generate(n, p, true_beta):
    X = np.random.normal(loc=0, scale=1, size=(n, p))
    true_beta = np.reshape(true_beta, (p, 1))

    true_y = np.dot(X, true_beta)
    noise = np.random.normal(loc=0, scale=1, size=(n, 1))
    y = true_y + noise
    return X, y, true_y
```

This is just the following:

$$\begin{aligned} X, \text{noise} &\sim N(0, I) \\ Y_{\text{true}} &= X\beta_{\text{true}} \\ Y &= Y_{\text{true}} + \text{noise} \end{aligned} \tag{21}$$

2.2 Utils

2.2.1 Construct S

```
def construct_s(bh):
    s = []
    for bhj in bh:
        if bhj != 0:
            s.append(np.sign(bhj))
    s = np.array(s)
    s = s.reshape((len(s), 1))
    return s
```

Input: β

Output: S

2.2.2 Construct $A, X_A, A_c, X_{A_c}, \beta_A$

```
def construct_A_XA_Ac_XAc_bhA(X, bh, p):
    A = []
    Ac = []
    bhA = []

    for j in range(p):
        bhj = bh[j]
        if bhj != 0:
            A.append(j)
            bhA.append(bhj)
        else:
            Ac.append(j)

    XA = X[:, A]
    XAc = X[:, Ac]
    bhA = np.array(bhA).reshape((len(A), 1))

    return A, XA, Ac, XAc, bhA
```

The name itself already explains what going on.

2.2.3 Construct Test Statistic (The name is still self-explainatory)

```
def construct_test_statistic(j, XA, y, A):
    ej = []
    for each_j in A:
        if j == each_j:
            ej.append(1)
        else:
            ej.append(0)
    ej = np.array(ej).reshape((len(A), 1))
    inv = np.linalg.pinv(np.dot(XA.T, XA))
    XAinv = np.dot(XA, inv)
    etaj = np.dot(XAinv, ej)
    etajTy = np.dot(etaj.T, y)[0][0]
    return etaj, etajTy
```

2.2.4 Caculate CDF for Truncated Normal Distribution

```
def pivot_with_specified_interval(z_interval, etaj, etajTy, cov, tn_mu):

    tn_sigma = np.sqrt(np.dot(np.dot(etaj.T, cov), etaj))[0][0]

    numerator = 0
    denominator = 0

    for each_interval in z_interval:
        al = each_interval[0]
        ar = each_interval[1]

        denominator = denominator + mp.ncdf((ar - tn_mu)/tn_sigma) - mp.ncdf((al -
tn_mu)/tn_sigma)

        if etajTy >= ar:
            numerator = numerator + mp.ncdf((ar - tn_mu)/tn_sigma) - mp.ncdf((al -
tn_mu)/tn_sigma)
        elif (etajTy >= al) and (etajTy < ar):
            numerator = numerator + mp.ncdf((etajTy - tn_mu)/tn_sigma) -
mp.ncdf((al - tn_mu)/tn_sigma)

    if denominator != 0:
        return float(numerator/denominator)
    else:
        return None
```

This is just multi-interval case of TN, let (a_i, b_i) for $i \in [1,k]$ and $\Phi(z) = \text{CDF of N(0,1)}$

$$\text{denominator} = \sum_{i=1}^{k}\left[\Phi\left(\frac{b_i-\mu}{\sigma}\right) - \Phi\left(\frac{a_i-\mu}{\sigma}\right)\right] \tag{22}$$

$$\text{numerator} = \begin{cases} 0 \text{ , if } x < a_1 \\ 1 \text{ , if } x > b_k \\ \sum_{i=1}^{j-1}\left[\Phi\left(\frac{b_i-\mu}{\sigma}\right) - \Phi\left(\frac{a_i-\mu}{\sigma}\right)\right] \text{ , if } a_j \le x \le b_j \text{ for } j \in [1,k] \\ \sum_{i=1}^{j}\left[\Phi\left(\frac{b_i-\mu}{\sigma}\right) - \Phi\left(\frac{a_i-\mu}{\sigma}\right)\right] \text{ , if } b_j < x < a_{j+1} \text{ for } j \in [1,k-1] \end{cases} \tag{23}$$

2.3 Main

I am too lazy to do this part, just check the code.

$$\beta_{\text{true}} = [0,0,0,0,0] \tag{24}$$

And we choose random entry in beta + noise and make inference about it, doing this for large amount of iteration, because

$$\mathbb{H}_{1,j} : \beta_j = 0 \tag{25}$$

Then the p-value plot should be $\sim \text{Uniform}(0,1)$