4. Simulation for Lasso and OLS Regression

November 20, 2020

0.1 4. Simulation for Lasso and OLS Regression

We consider the multivariate regression model

$$Y = X\beta + \varepsilon = \sum_{j=1}^{p} \beta_j X_j + \varepsilon_i,$$

where $X = (X_1, \ldots, X_p) \in \mathbb{R}^p$ is a p-dimensional vector of regressors and β a p-dimensional vector of regression coefficients. The β-vector is *sparse*, i.e., only the first s elements of the vector are different from zero, and all others are exactly equal to zero.

1. In your own words, explain the idea behind Lasso, Ridge and Elastic Net and in what respect they differ from ordinary least squares.
2. Implement a data generating process according to the following setting:
   - $n = 100$,
   - $p = 50$,
   - $s = 5$,
   - $\beta = (5, 4, 3, 2, 1, 0, 0, \ldots, 0)$
   - $\varepsilon \sim N(0, 1)$
   - The covariates are drawn from a multivariate normal distribution with a $(p \times p)$-covariance matrix

$$\Sigma = \begin{pmatrix} 1 & 0.25 & ... & 0.25 \\ 0.25 & 1 & \ldots & 0.25 \\ \ldots & \ldots & \ldots & \ldots \\ 0.25 & 0.25 & \ldots & 1 \end{pmatrix}$$

   (all entries in the covariance matrix are equal to 0.25 except for those on the diagonal. These values are equal to 1.)
   Hint: It will be useful if you implement the data generating process in a function.
3. Generate a sample according to the data generating process above. Estimate the regression coefficients by ordinary least squares and lasso regression. Calculate and compare the squared error, i.e., $(\hat{\beta}_j - \beta_j)^2$ for $j = 1, 5, 50$ for the lasso and ols estimators.

4. Perform a simulation study, i.e., repeat your calculation from part 3. $R = 1000$ times. Calculate the mean squared error for the lasso and ols coefficients for the first, fifth, and 50th regressor, i.e., average the squared error over the R repetitions. For the first, fifth, and 50th regressor, how many times is the corresponding coefficient (estimated with lasso) set exactly to zero?

5. How do your result in part 4. change, if you increase the number of regressors p to 70, 90, and, 110? Illustrate your findings with an appropriate figure and summarize your results.

0.1.1 4. Simulation for Lasso and OLS Regression

[2]:
```
from IPython.core.interactiveshell import InteractiveShell     #allows printing␣
↪multiple lines
InteractiveShell.ast_node_interactivity = "all"
```

[3]:
```
#def print(*args):
#    __builtins__.print(*("%.4f" % a if isinstance(a, float) else a
#                         for a in args))
```

1. Explain Lasso, Ridge and Elastic Net and how they differ from OLS:

2. Implement data generating process

[4]:
```
import numpy as np

#parameter
n= 100
p=70
s=5
mu = np.array([0]*p)            #oder np.zeros(p)

#covariance matrix
cov = np.linspace(0.25,0.25, p*p)      #np.identity(p) creates an identity␣
↪matrix with 0s and 1s on the diagonal
cov =np.array(cov)
cov = cov.reshape(p,p)
np.fill_diagonal(cov, 1.)

#coefficient
null = np.array([0]*(p-5))
coef=np.array([5,4,3,2,1])
coef=np.concatenate((coef,null))
```

[193]:
```
cov
cov.shape
coef
coef.shape
mu.shape
```

[193]:
```
array([[1.  , 0.25, 0.25, …, 0.25, 0.25, 0.25],
       [0.25, 1.  , 0.25, …, 0.25, 0.25, 0.25],
       [0.25, 0.25, 1.  , …, 0.25, 0.25, 0.25],
       …,
       [0.25, 0.25, 0.25, …, 1.  , 0.25, 0.25],
       [0.25, 0.25, 0.25, …, 0.25, 1.  , 0.25],
       [0.25, 0.25, 0.25, …, 0.25, 0.25, 1.  ]])
```

[193]:
```
(70, 70)
```

[193]:
```
array([5, 4, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0])
```

[193]:
```
(70,)
```

[193]:
```
(70,)
```

[16]:
```
#create function for generating a sample

def gen_sample(mu, n, p):
    cov = np.linspace(0.25,0.25, p*p)
    cov =np.array(cov)
    cov = cov.reshape(p,p)
    np.fill_diagonal(cov, 1.)
    X = np.random.multivariate_normal(mu, cov, n)
    error = np.random.normal(loc=0.0, scale=1.0, size=n)
    y = np.matmul(X, coef)          #oder np.dot(x, beta) (yield different␣
 ↪results when mult. 3D matrices)
    y = np.add(y, error)
    return X,y
```

[17]:
```
#generate a sample

X, y = gen_sample(mu, n, p)
```

0.1.2 Regularisierung

Estimate coefficients by linear regression and lasso

[196]:
```
#linear regression

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X, y)
#print("Coefficient:", model.coef_)
coef_lin_reg = model.coef_
```

```
#coef_lin_reg.shape
```

[196]:
```
LinearRegression()
```

[21]:
```
#lasso with CV (better MSE)

from sklearn.linear_model import LassoCV

model = LassoCV()
model.fit(X, y)
print("Coefficient:", model.coef_)
coef_lasso = model.coef_
#coef_lasso.shape
```

[21]:
```
LassoCV()
```

```
Coefficient: [ 4.97791955e+00  3.85640096e+00  2.85836706e+00  1.85442840e+00
  7.28051551e-01  0.00000000e+00 -0.00000000e+00  0.00000000e+00
  0.00000000e+00  5.84021485e-02  0.00000000e+00 -0.00000000e+00
  0.00000000e+00  2.30696803e-03  0.00000000e+00  6.93416160e-02
  9.92131197e-03  0.00000000e+00 -0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00  4.63803507e-02  0.00000000e+00 -0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  3.50509793e-02  3.67404155e-04  0.00000000e+00  0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  2.90228889e-02
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
  9.18624260e-02  5.05358243e-02  0.00000000e+00  0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
 -0.00000000e+00  0.00000000e+00  0.00000000e+00  1.09903656e-01
  0.00000000e+00  0.00000000e+00]
```

[28]:
```
from sklearn import linear_model

#lasso without CV

model = linear_model.Lasso(alpha=1, fit_intercept=True)
model.fit(X, y)
print("Coefficient:", model.coef_)
print("Coefficient:", model.sparse_coef_)

#coef_lasso = model.coef_
#coef_lasso.shape
```

[28]: Lasso(alpha=1)

```
Coefficient: [4.66246712 3.38033257 2.61354245 1.37123858 0.25023747 0.
 0.         0.19165049 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
Coefficient:   (0, 0)   4.662467119648517
  (0, 1)        3.3803325748035613
  (0, 2)        2.6135424485226797
  (0, 3)        1.3712385764438566
  (0, 4)        0.2502374704928779
  (0, 7)        0.19165049147449337
```

[198]:
```
#squared error for j=1,5,50

error_lin_reg = (coef_lin_reg-coef)**2
error_lasso = (coef_lasso-coef)**2

print("Error for j=1", np.c_[error_lin_reg[0], error_lasso[0]])
print("Error for j=5",np.c_[error_lin_reg[4], error_lasso[4]])
print("Error for j=50",np.c_[error_lin_reg[-1], error_lasso[-1]])
#print("\n")
#print("Total Error for j= 1,...,50", np.c_[np.sum(error_lin_reg), np.
 ↪sum(error_lasso)])
```

```
Error for j=1 [[0.01421365 0.04925789]]
Error for j=5 [[0.02212844 0.06805104]]
Error for j=50 [[0.0371114 0.       ]]
```

Lasso has lower total error than linear regression

[199]:
```
#create function for comparing the squared error and outputting lasso␣
 ↪coefficients

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LassoCV

def compare_squared_error(mu, n, p):
    k=100
    error_lin_reg1_all =[]
```

```
    error_lin_reg5_all =[]
    error_lin_reg50_all =[]

    error_lasso1_all =[]
    error_lasso5_all =[]
    error_lasso50_all =[]

    lasso_coef1 =[]
    lasso_coef5 =[]
    lasso_coef50 =[]

    for i in range(k):
        X, y = gen_sample(mu, n, p)
        model1 = LinearRegression()
        model1.fit(X, y)
        error_lin_reg1 = (model1.coef_[0]-coef[0])**2
        error_lin_reg5 = (model1.coef_[4]-coef[4])**2
        error_lin_reg50 = (model1.coef_[-1]-coef[-1])**2

        model2 = LassoCV()
        model2.fit(X, y)
        error_lasso1 = (model2.coef_[0]-coef[0])**2
        error_lasso5 = (model2.coef_[4]-coef[4])**2
        error_lasso50 = (model2.coef_[-1]-coef[-1])**2

        #save OLS and lasso errors
        error_lin_reg1_all.append(error_lin_reg1)
        error_lin_reg5_all.append(error_lin_reg5)
        error_lin_reg50_all.append(error_lin_reg50)

        error_lasso1_all.append(error_lasso1)
        error_lasso5_all.append(error_lasso5)
        error_lasso50_all.append(error_lasso50)

        #save lasso coefficients
        lasso_coef1.append(model2.coef_[0])
        lasso_coef5.append(model2.coef_[4])
        lasso_coef50.append(model2.coef_[-1])

    return error_lin_reg1_all, error_lin_reg5_all, error_lin_reg50_all,␣
↪error_lasso1_all, error_lasso5_all, error_lasso50_all, lasso_coef1,␣
↪lasso_coef5, lasso_coef50
```

[200]:
```
import time

t0 = time.time()
```

```
error_lin_reg1_all, error_lin_reg5_all, error_lin_reg50_all, error_lasso1_all,␣
↪error_lasso5_all, error_lasso50_all, lasso_coef1, lasso_coef5, lasso_coef50␣
↪= compare_squared_error(mu, n, p)
t1 = time.time()
total = t1-t0
print("Run Time:", total)
```

[200]:
```
22.293280124664307
```

[201]:
```
error_lin_reg1_all[:5]
len(error_lin_reg1_all)

lasso_coef1[:5]
lasso_coef5[:5]
lasso_coef50[:5]

len(lasso_coef1)
```

[201]:
```
[0.021018105085882692,
 0.0025004581556362696,
 0.037041761407783334,
 0.001390201662060141,
 0.004090714204050163]
```

[201]:
```
100
```

[201]:
```
[5.023094351774674,
 4.811340322501862,
 4.894358965724214,
 4.915650180234686,
 4.854756710737661]
```

[201]:
```
[0.637503339680049,
 0.8688391006706642,
 0.8756029132338973,
 0.7377563175793532,
 1.0044372463542424]
```

[201]:
```
[-0.0, 0.07626736190594183, -0.019774431347079353, 0.0, 0.0]
```

[201]:
```
100
```

[202]:
```
mse_ols_1 = np.mean(error_lin_reg1_all)
mse_ols_5 = np.mean(error_lin_reg5_all)
mse_ols_50 = np.mean(error_lin_reg50_all)

mse_lasso_1 = np.mean(error_lasso1_all)
```

```
mse_lasso_5 = np.mean(error_lasso5_all)
mse_lasso_50 = np.mean(error_lasso50_all)
```

[203]:
```
print("Compare Errors:", "OLS - Lasso")

mse_ols_1 - mse_lasso_1

mse_ols_5 - mse_lasso_5

mse_ols_50 - mse_lasso_50

print("The OLS-error is lower for the first two coefficients, while the␣
 ↪lasso-error is lower for the last sparce coefficient")
```

```
Compare Errors: OLS - Lasso
```

[203]:
```
0.010282940241642051
```

[203]:
```
0.003512165071154972
```

[203]:
```
0.05431223028384472
```

```
The OLS-error is lower for the first two coefficients, while the lasso-error is
lower for the last sparce coefficient
```

[204]:
```
list =[lasso_coef1[:5], lasso_coef5[:5],lasso_coef50[:5]]

for a in zip(*list):
    print(*a)
```

```
5.0231 0.6375 -0.0000
4.8113 0.8688 0.0763
4.8944 0.8756 -0.0198
4.9157 0.7378 0.0000
4.8548 1.0044 0.0000
```

[205]:
```
#count how many times the lasso coefficient is set exactly to zero

count_1 = 0
count_5 = 0
count_50 = 0

for i in lasso_coef1:
    if i == 0:
        count_1 += 1
for i in lasso_coef5:
    if i == 0:
        count_5 += 1
```

```
for i in lasso_coef50:
    if i == 0:
        count_50 += 1

print(count_1)
print(count_5)
print(count_50)


#error_lasso1_all.shape
#print(np.round(error_lasso1_all, 4))
```

```
0
0
83
```

The coefficient of the 50th regressor is 70 out of the 100 times exactly set to zero.

0.1.3 Muss das Process automatisieren, (Funktionen einbauen) und eine Figure (vorzüglich math.plot.lib) mit den Ergebnissen