tags: 資料科學計算

Homework assignments Week 11 - Programming Work

Due date: 2021/12/11
Author: Hung Chun Hsu
Student ID: R10946017
線上閱讀: https://hackmd.io/@stupid-penguin/r1Z1ZYW5K (https://hackmd.io/@stupid-penguin/r1Z1ZYW5K)
Github Repository: https://github.com/stupidpenguin/Computation_of_Data_Science/tree/master/Homework_5 (https://github.com/stupidpenguin/Computation_of_Data_Science/tree/master/Homework_5)

- For the coding details, please see the attached file -> “2_The_ridge_regression_estimation.ipynb”, thanks!

Table of Contents

2. The ridge regression estimation (6%)

2-1. Randomly generate a dataset

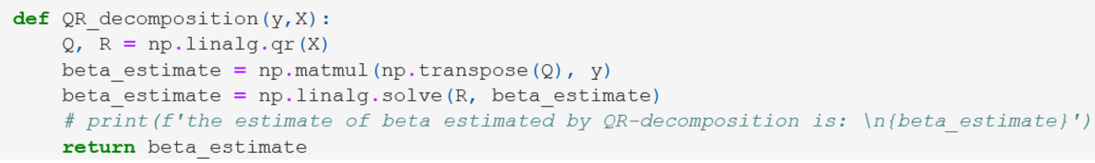
```python
def Initialization(n, p):
    beta_true = np.ones((p, 1))
    X = np.random.normal(0, 1, size=(n,p))
    noise = np.random.normal(0, 1, size=(n, 1))
    # print(f'beta_ture shape: {beta_true.shape}')
    # print(X.shape)
    # print(noise.shape)
    y = np.matmul(X, beta_true)+noise
    # print(y)
    # print(y.shape)
    return y, X
```

2-2. Computing the estimate with algorithms

a. Cholesky-BFS

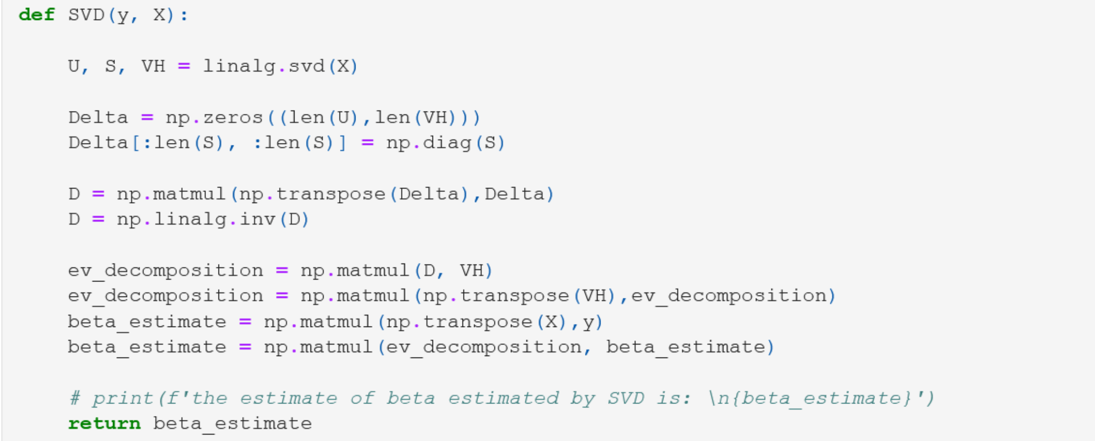
```python
def Cholesky(y, X):
    L = np.linalg.cholesky(np.matmul(np.transpose(X),X))
    # print(L)

    theta = np.matmul(np.transpose(X), y)
    theta = np.linalg.solve(L, theta)

    beta_estimate = np.linalg.solve(np.transpose(L),theta)
    # print(f'the estimate of beta estimated by Cholesky-BFS is: \n{beta_estimate}')
    return beta_estimate
```

b. QR decomposition



```
def QR_decomposition(y,X):
    Q, R = np.linalg.qr(X)
    beta_estimate = np.matmul(np.transpose(Q), y)
    beta_estimate = np.linalg.solve(R, beta_estimate)
    # print(f'the estimate of beta estimated by QR-decomposition is: \n{beta_estimate}')
    return beta_estimate
```

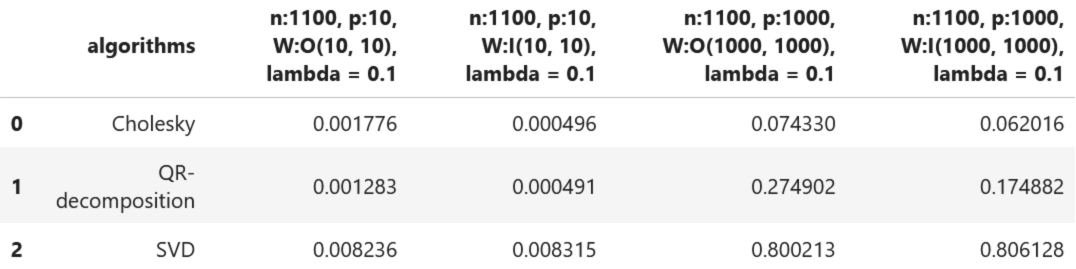
c. SVD-based algorithm (based on Problem 1.)



```
def SVD(y, X):

    U, S, VH = linalg.svd(X)

    Delta = np.zeros((len(U),len(VH)))
    Delta[:len(S), :len(S)] = np.diag(S)

    D = np.matmul(np.transpose(Delta),Delta)
    D = np.linalg.inv(D)

    ev_decomposition = np.matmul(D, VH)
    ev_decomposition = np.matmul(np.transpose(VH),ev_decomposition)
    beta_estimate = np.matmul(np.transpose(X),y)
    beta_estimate = np.matmul(ev_decomposition, beta_estimate)

    # print(f'the estimate of beta estimated by SVD is: \n{beta_estimate}')
    return beta_estimate
```

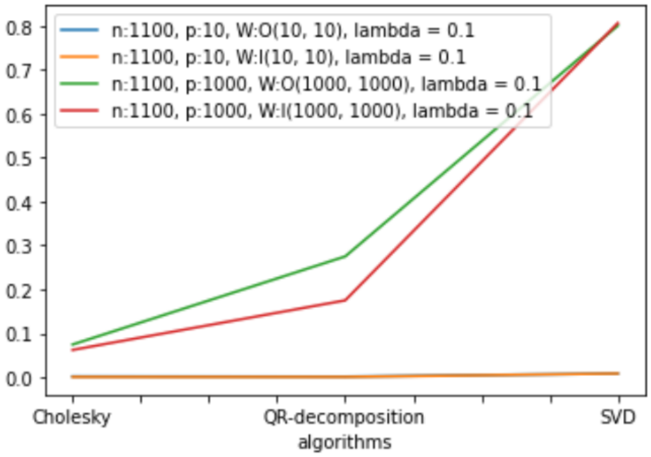
2-3. Runtime of the three algorithms

a. Comparison of Runtime (seconds)



	algorithms	n:1100, p:10, W:O(10, 10), lambda = 0.1	n:1100, p:10, W:I(10, 10), lambda = 0.1	n:1100, p:1000, W:O(1000, 1000), lambda = 0.1	n:1100, p:1000, W:I(1000, 1000), lambda = 0.1
0	Cholesky	0.001776	0.000496	0.074330	0.062016
1	QR-decomposition	0.001283	0.000491	0.274902	0.174882
2	SVD	0.008236	0.008315	0.800213	0.806128

b. Line Chart of the result


n:1100, p:10, W:O(10, 10), lambda = 0.1
n:1100, p:10, W:I(10, 10), lambda = 0.1
n:1100, p:1000, W:O(1000, 1000), lambda = 0.1
n:1100, p:1000, W:I(1000, 1000), lambda = 0.1
Cholesky
QR-decomposition
SVD
algorithms