R10946013 劉謦瑄

Homework assignments Week 11 (Students should submit their homework before 10 a.m. on December 8, 2021.)

1. Computing the least squares estimate via SVD (2%)

Consider the least squares estimate

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{1}$$

where $\mathbf{X}$ is an $n \times p$ matrix with $n \geq p$ and $\mathbf{y}$ is an n-dimensional vector. Below we provide a way of using the singular value decomposition to compute (1). First note that if $\mathbf{X}$ has the singular value decomposition $\mathbf{X} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^T$, where $\mathbf{U}$ is an $n \times n$ matrix, $\boldsymbol{\Lambda}$ is an $n \times p$ matrix, and $\mathbf{V}$ is a $p \times p$ matrix. Then which of the following statements are true?

- **My Answer: (b).**

b. $\mathbf{V}\mathbf{D}\mathbf{V}^T$ is the eigenvalue decomposition of $(\mathbf{X}^T\mathbf{X})^{-1}$, where $\mathbf{D} = (\boldsymbol{\Lambda}^T\boldsymbol{\Lambda})^{-1}$ is a $p \times p$ diagonal matrix.

Programming work

2. The ridge regression estimation (6%)

Now consider the following estimate:

$$\widehat{\boldsymbol{\beta}}^{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \frac{\lambda}{2}||\mathbf{W}\boldsymbol{\beta}||_2^2 \right\},$$

where $\mathbf{y}$ is an n-dimensional vector, $\mathbf{X}$ is an $n \times p$ matrix, $\boldsymbol{\beta}$ is a p-dimensional vector, $\lambda \geq 0$ is a scalar, and $\mathbf{W}$ is a $p \times p$ matrix. Here $\widehat{\boldsymbol{\beta}}^{\text{ridge}}$ is called the ridge estimate of regression coefficients $\boldsymbol{\beta}$. Our goal is to evaluate performance of (a) the Cholesky forward backward substitution, (b) QR decomposition-based algorithm and (c) SVD-based algorithm (see **Problem 2.** for details) for computing the ridge estimate $\widehat{\boldsymbol{\beta}}^{\text{ridge}}$.

We evaluate performances of the three algorithms by conducting simulation experiments. The experiment steps are given as follows:

1. Let $(\boldsymbol{\beta}^{\text{true}})_j = 1$ for $j = 1, 2, \cdots, p$. Generate data by first drawing $(\mathbf{X})_{ij} = x_{ij}$ from Normal$(0, 1)$ for $i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, p$ and then computing response vector $\mathbf{y}$ by

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^{\text{true}} + \boldsymbol{\epsilon},$$

where $(\boldsymbol{\epsilon})_i \sim \text{Normal}(0, 1)$ for $i = 1, 2 \cdots, n$.

Reformulate as a least squares estimation problem: Let

$$\mathbf{z} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} \quad \text{and } \mathbf{U} = \begin{bmatrix} \mathbf{X} \\ -\sqrt{\lambda}\mathbf{W} \end{bmatrix}$$

where z is an (n + p)-dimensional vector and U is an (n + p) × p matrix. Then the ridge estimator can be expressed as:

$$\widehat{\boldsymbol{\beta}}^{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} ||\mathbf{z} - \mathbf{U}\boldsymbol{\beta}||_2^2 \right\}$$

e.g. For my experiment, while n=1100, p=10, W is zero metrix:

```
z:
 [1.71509052 0.38259422 5.91640191 ... 0.          0.          0.        ]
```

```
U:
 [[-1.74976547  0.3426804   1.1530358  ... -1.07004333 -0.18949583
   0.25500144]
 [-0.45802699  0.43516349 -0.58359505 ...  1.02973269 -0.43813562
  -1.11831825]
 [ 1.61898166  1.54160517 -0.25187914 ...  1.36155613 -0.32623806
   0.05567601]
 ...
 [ 0.          0.          0.         ...  0.          0.
   0.        ]
 [ 0.          0.          0.         ...  0.          0.
   0.        ]
 [ 0.          0.          0.         ...  0.          0.
   0.        ]]
```

2. Run the following three algorithms for computing $\widehat{\boldsymbol{\beta}}^{\text{ridge}}$:

(a) The Cholesky forward backward substitution (Cholesky-FBS);

(b) The QR decomposition-based algorithm (QR);

(c) The SVD-based algorithm (SVD-based; See **Problem 1.** for details).

(a)

```
#cholesky factorization
L = linalg.cholesky(S, lower=True)
#check LL.T  = S
np.dot(L, L.T)
# solve for the coefficents
theta = linalg.solve_triangular(L, np.dot(X.T, y), lower=True)
beta_chol = linalg.solve_triangular(L.T, theta, lower=False)
```

```
beta_chol : [1.03277018 1.03027686 0.98263862 1.01411532 1.04074975 0.97569903
 0.98130359 1.0094818  0.99676169 1.03289357]
```

(b)

```
Q, R = linalg.qr(X, mode = 'economic')
beta_qr = np.linalg.inv(R).dot(Q.T).dot(y)
```

```
beta_qr : [1.03277018 1.03027686 0.98263862 1.01411532 1.04074975 0.97569903
 0.98130359 1.0094818  0.99676169 1.03289357]
```

(c)

```
U, Sig, Vt = linalg.svd(X, full_matrices=False)
z = (1 / Sig) * U.T.dot(y)
beta_svd = Vt.T.dot(z)
```

```
beta_svd : [1.03277018 1.03027686 0.98263862 1.01411532 1.04074975 0.97569903
 0.98130359 1.0094818  0.99676169 1.03289357]
```

3. Record the runtime of the above three algorithms.

- The runtime of the three algorithms are Reported in "seconds" for experiments:

(n, p, W, λ)	Cholesky-FBS	QR-based	SVD-based
(1100, 10, 0_{pxp}, 0.1)	**0.0007998943328857422**	**0.0009379386901855469**	**0.03258180618286133**
(1100, 1000, 0_{pxp}, 0.1)	**0.14247989654541016**	**0.451016902923584**	**1.0703041553497314**
(1100, 10, I_{pxp}, 0.1)	**0.0010671615600585938**	**0.0013148784637451172**	**0.04789113998413086**
(1100, 1000, I_{pxp}, 0.1)	**0.13515210151672363**	**0.4830000400543213**	**1.1729578971862793**