STAT 545 HW 4

2026-02-09

1

Consider a regression model with p < n:

$$y = X\beta + \epsilon$$

where $y \in R^n, X \in R^{n\times p}$ and $\epsilon \sim N(0, \sigma^2 I_n)$ and rank(X) = p. Recall that the ridge regression estimate is given by

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

for given $\lambda > 0$.

A.

Let the SVD of X be given by $X = UDV^T$. Show that the fitted value is

$$\hat{y} = X\hat{\beta}_{ridge} = \sum_{j=1}^{p} u_j \frac{d_j^2}{d_j^2 + \lambda} u_j^T y$$

where u_j are the columns of U and $D = diag(d_j)$.

First, since it is SVD, we know that $U^T U = V^T V = I_p$. We know the columns of U and V form an orthonormal bases for the row and column space of X respectively.

We first want to calculate $\hat{\beta}_{ridge}$.

We know from chapter 2 notes, we can find

$$X^T X = (VDU^T)(UDV^T) = VD^2V^T$$

Then, we can get it in a nice form like in the slides

$$(X^T X + \lambda I) = VD^2V^T + \lambda(VV^T) = V(D^2 + \lambda I)V^T$$

Then, simple inverse (Is possible because we added lambda values along diagonal, making it positive definite). Note that the inverse of an orthonormal matrix is its own transpose.

$$(X^T X + \lambda I)^{-1} = (V(D^2 + \lambda I)V^T)^{-1} = V(D^2 + \lambda I)^{-1}V^T$$

We can now plug everything in and find

$$
\begin{aligned}
\hat{\beta}_{ridge} &= V(D^2+\lambda I)^{-1}V^T(VDU^T)y \\
&= V(D^2+\lambda I)^{-1}DU^Ty
\end{aligned}
$$

Multiply by X now.

$$
\begin{aligned}
\hat{y} &= X\hat{\beta}_{ridge} \\
&= (UDV^T)V(D^2+\lambda I)^{-1}DU^Ty \\
&= UD(D^2+\lambda I)^{-1}DU^Ty \\
&= \sum_{j=1}^{p} u_j \frac{d_j^2}{d_j^2+\lambda} u_j^T y
\end{aligned}
$$

Each u_j is a column vector multiplied by the value of the j-th diagonal value in D (a diagonal matrix, $p \times p$), so we can treat $D(D^2+\lambda I)^{-1}D$ as $\frac{d_j^2}{d_j^2+\lambda}$ and split the calculation of $\hat{y}$ as a sum of the operations on the j columns.

B.

When the fitted value $\hat{y}$ has a form $\hat{y} = My$ where M is a matrix that does not depend on y then the "degrees of freedom" is given by the trace of the matrix M. Derive an expression (a scalar quantity) for the degrees of freedom of ridge regression.

We know

$$
\begin{aligned}
\hat{y} &= X\hat{\beta}_{ridge} \\
&= (UDV^T)V(D^2+\lambda I)^{-1}DU^Ty \\
&= UD(D^2+\lambda I)^{-1}DU^Ty \\
\implies M &= UD(D^2+\lambda I)^{-1}DU^T
\end{aligned}
$$

Then

$$
\begin{aligned}
tr(M) &= tr(UD(D^2+\lambda I)^{-1}DU^T) \\
&= tr(D(D^2+\lambda I)^{-1}DU^TU) \\
&= tr(D(D^2+\lambda I)^{-1}D) \\
&= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2+\lambda}
\end{aligned}
$$

Given that the degree of freedom is equal to the trace, we find that $df = tr(M) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2+\lambda}$.

C.

Take n = 70, p = 100. Simulate X and ϵ from i.i.d. standard normals. Take β to be a vector of all ones and set $y = X\beta + \epsilon$. Take $\lambda = 1$ and verify that (a) that the ridge estimate is well-defined and (b) both ways of calculating the fitted value above (via matrix inverse and via SVD) indeed result in the same $\hat{y}$ in R. The commands svd() for calculating SVD may be useful.

First we set it all up.

```r
set.seed(13)
n <- 70
p <- 100
X <- matrix(rnorm(n * p), nrow = n, ncol = p)
epsilon <- rnorm(n)
beta <- rep(1, p)
y <- X %*% beta + epsilon
lambda <- 1
```

Direct computation way

```r
beta_ridge <- solve(t(X) %*% X + lambda * diag(p)) %*% t(X) %*% y
y_hat_inv <- X %*% beta_ridge

# Check values
beta_ridge
```

```
##                [,1]
##    [1,]  1.253267328
##    [2,]  0.824068812
##    [3,]  0.522741183
##    [4,]  0.978814764
##    [5,]  0.466439308
##    [6,]  1.250124085
##    [7,]  0.321746645
##    [8,]  1.240042943
##    [9,]  0.319374310
##   [10,]  0.979175475
##   [11,]  0.665154813
##   [12,]  1.899374467
##   [13,]  0.414776278
##   [14,]  1.486085861
##   [15,]  0.769178241
##   [16,]  0.682495639
##   [17,]  0.312515779
##   [18,]  0.608304003
##   [19,]  1.090411287
##   [20,]  0.647268079
##   [21,]  1.935008401
##   [22,]  0.774677535
##   [23,]  0.503690748
##   [24,]  1.576099086
##   [25,]  0.883043617
##   [26,]  0.436607704
##   [27,]  0.902727906
##   [28,]  1.402553778
##   [29,]  1.090050344
##   [30,]  0.874637508
##   [31,]  0.990564358
##   [32,]  0.376301286
##   [33,]  1.066534515
```

```
## [34,] -0.060808858
## [35,]  0.133728141
## [36,]  0.591757043
## [37,]  0.938083854
## [38,]  1.153184984
## [39,] -0.003031669
## [40,]  0.857557656
## [41,] -0.300039862
## [42,]  0.670482295
## [43,]  0.863936516
## [44,]  0.093572456
## [45,]  0.371717743
## [46,]  0.833470563
## [47,]  0.807508827
## [48,]  1.036199830
## [49,]  1.844007927
## [50,]  1.116752435
## [51,]  0.984484274
## [52,]  0.682835724
## [53,]  0.775713528
## [54,]  0.850923473
## [55,]  0.479990672
## [56,]  0.144934484
## [57,]  0.533261128
## [58,]  1.636666867
## [59,]  0.200729498
## [60,]  0.422847735
## [61,]  1.058942156
## [62,]  0.282593915
## [63,]  0.948986634
## [64,]  0.619825939
## [65,]  0.626653348
## [66,]  0.317341709
## [67,]  0.473644646
## [68,]  0.536327016
## [69,] -0.104662967
## [70,] -0.050013051
## [71,]  1.021453146
## [72,]  0.298307614
## [73,]  0.776355498
## [74,]  0.663582287
## [75,]  0.720490266
## [76,]  0.626932288
## [77,]  1.114055982
## [78,]  1.449874503
## [79,]  0.683645985
## [80,] -0.102013800
## [81,]  1.494683561
## [82,]  0.307068367
## [83,]  0.468604824
## [84,]  0.787521556
## [85,]  1.284092059
## [86,]  0.948890748
## [87,]  0.407425831
```

```
##  [88,]  0.630615860
##  [89,]  1.184312322
##  [90,]  0.453168164
##  [91,]  0.447722957
##  [92,]  0.720061814
##  [93,]  0.651741818
##  [94,]  0.790570056
##  [95,]  0.888219252
##  [96,]  0.726008138
##  [97,]  0.616305138
##  [98,]  0.928586003
##  [99,]  0.734734715
## [100,]  0.075364795
```

```
y_hat_inv
```

```
##              [,1]
##  [1,]  11.6930595
##  [2,]  -2.2828304
##  [3,]   8.3389511
##  [4,] -17.3525478
##  [5,]   0.7857293
##  [6,]  -3.0377488
##  [7,]   9.8427591
##  [8,]   8.9135213
##  [9,]   2.0486252
## [10,]  -1.2649932
## [11,]  11.4920223
## [12,] -15.9795011
## [13,]  -5.0582121
## [14,]  -3.0245648
## [15,]  -1.3977781
## [16,]  -3.7033322
## [17,]   8.0426574
## [18,] -23.7889261
## [19,]  -4.6927534
## [20,]  16.1170924
## [21,]   4.6581993
## [22,]   6.2699274
## [23,]  -8.3129443
## [24,]  10.8168377
## [25,] -14.1247429
## [26,]   2.8640045
## [27,] -20.9264941
## [28,]  -3.3688354
## [29,]   2.1126005
## [30,] -20.6679985
## [31,]   6.6283992
## [32,]  17.8736022
## [33,] -21.2346204
## [34,]   6.2344835
## [35,] -11.2643976
## [36,]   5.6457907
## [37,]  13.6744408
```

```
## [38,] -11.9604539
## [39,]   1.3115843
## [40,] -16.1456468
## [41,] -14.2352921
## [42,]  10.0227490
## [43,]  -7.8865110
## [44,]   7.9407564
## [45,] -16.8534783
## [46,]  22.0800211
## [47,] -17.7969466
## [48,]  13.1852053
## [49,]  -7.5489433
## [50,]   8.7510592
## [51,]  11.5745645
## [52,]   9.4745742
## [53,]  -5.5360794
## [54,]   3.6792100
## [55,]  -0.7659997
## [56,]   4.4279193
## [57,]   4.9919759
## [58,]  17.4007937
## [59,]   6.8553628
## [60,]   2.3306133
## [61,]  -6.4441001
## [62,]   9.8295794
## [63,]  11.8660865
## [64,]   6.0520626
## [65,] -20.8839999
## [66,]   8.2596429
## [67,]   1.3960509
## [68,]   4.9397570
## [69,]   3.0433631
## [70,] -16.3029813
```

The SVD way

We find β_{ridge} and then compute the estimate $\hat{y}$.

```
SVD_X <- svd(X)
SVD_U <- SVD_X$u
SVD_D <- SVD_X$d

# We use the formula from part A.
SVD_y_hat <- matrix(0, nrow = n, ncol = 1)
for (j in 1:length(SVD_D)) {
  uj <- SVD_U[, j, drop = FALSE]
  d_diagonals <- (SVD_D[j]^2) / (SVD_D[j]^2 + lambda)
  UTY <- t(uj) %*% y
  SVD_y_hat <- SVD_y_hat + uj %*% (d_diagonals * UTY)
}

# Check our results
print(SVD_y_hat)
```

```
##              [,1]
##  [1,]  11.6930595
##  [2,]  -2.2828304
##  [3,]   8.3389511
##  [4,] -17.3525478
##  [5,]   0.7857293
##  [6,]  -3.0377488
##  [7,]   9.8427591
##  [8,]   8.9135213
##  [9,]   2.0486252
## [10,]  -1.2649932
## [11,]  11.4920223
## [12,] -15.9795011
## [13,]  -5.0582121
## [14,]  -3.0245648
## [15,]  -1.3977781
## [16,]  -3.7033322
## [17,]   8.0426574
## [18,] -23.7889261
## [19,]  -4.6927534
## [20,]  16.1170924
## [21,]   4.6581993
## [22,]   6.2699274
## [23,]  -8.3129443
## [24,]  10.8168377
## [25,] -14.1247429
## [26,]   2.8640045
## [27,] -20.9264941
## [28,]  -3.3688354
## [29,]   2.1126005
## [30,] -20.6679985
## [31,]   6.6283992
## [32,]  17.8736022
## [33,] -21.2346204
## [34,]   6.2344835
## [35,] -11.2643976
## [36,]   5.6457907
## [37,]  13.6744408
## [38,] -11.9604539
## [39,]   1.3115843
## [40,] -16.1456468
## [41,] -14.2352921
## [42,]  10.0227490
## [43,]  -7.8865110
## [44,]   7.9407564
## [45,] -16.8534783
## [46,]  22.0800211
## [47,] -17.7969466
## [48,]  13.1852053
## [49,]  -7.5489433
## [50,]   8.7510592
## [51,]  11.5745645
## [52,]   9.4745742
## [53,]  -5.5360794
```

```
## [54,]   3.6792100
## [55,]  -0.7659997
## [56,]   4.4279193
## [57,]   4.9919759
## [58,]  17.4007937
## [59,]   6.8553628
## [60,]   2.3306133
## [61,]  -6.4441001
## [62,]   9.8295794
## [63,]  11.8660865
## [64,]   6.0520626
## [65,] -20.8839999
## [66,]   8.2596429
## [67,]   1.3960509
## [68,]   4.9397570
## [69,]   3.0433631
## [70,] -16.3029813
```

Check results

```r
max(abs(y_hat_inv - SVD_y_hat))
```

```
## [1] 5.631051e-13
```

The two predictions that we got through the two different methods actually give us very similar results. They are nearly exactly the same.

We have a totally good prediction for $\hat{y}$ even though in theory, it is not possible to get the inverse of $X^T X$ for regular LS approach but with our ridge regression, we can still calculate some sort of inverse that is close and get predicted values.