Exercise set 2

your candidate number

High-dimensional regression

Background reading. Delete this section before submitting Wikipedia: Singular value decomposition. Read the introduction, Example, and Pseudoinverse sections.

Wikipedia: Pseudoinverse. Read the introduction and sections on Projectors, Examples, Linearly independent rows, and Applications.

ISLR: See section 12.5.2 for some example use of the `svd` function.

Delete from here to the previous line about deletion.

```
set.seed(1) # change this to some other number
# Generate a matrix of predictors X
# with 3 rows, 10 columns, and i.i.d. normal entries
p <- 10
n <- 3
X <- matrix(rnorm(n * p), nrow = n)
# Create a sparse coefficient vector beta
# with only 1 or 2 nonzero entries
beta <- rep(0, p)
beta[1] <- 1
# Compute outcome y from the noise-free linear model
y <- X %*% beta
```

Generate data (3 points)

Compute pseudoinverse (3 points) Use output from the `svd` function to compute a right pseudoinverse of `X`. (Note: if you use a source aside from the Wikipedia articles above to figure out how to do this you should cite your source and include a link if it's a website)

```
S <- svd(X)
X_pseudoinv <- S$v %*% diag(1/S$d) %*% t(S$u)
```

```
X %*% X_pseudoinv
```

Verify right-inverse property (3 points)

```
##               [,1]          [,2]          [,3]
## [1,]  1.000000e+00 -1.662692e-16 -1.289275e-16
## [2,] -3.247361e-16  1.000000e+00  1.401532e-16
## [3,] -1.886456e-16  3.672968e-16  1.000000e+00
```

Explanation: (1 point) Expected to be the identity matrix, and the diagonal entries are numerically close to 1 and off diagonal close to zero, i.e. the matrix is numerically close to the identity matrix.

Note: delete this comment after reading. If you are unable to compute the pseudoinverse using the svd function, you can increase the sample size to p+1 and use OLS to estimate beta instead to receive partial credit.

```r
beta_hat <- X_pseudoinv %*% y
cbind(beta, beta_hat) |> kable()
```

Compare estimated beta to true beta (3 points)

beta	
1	0.2152325
0	-0.1195434
0	-0.1159438
0	0.0664787
0	-0.1588557
0	-0.1190181
0	-0.2192990
0	0.1476748
0	-0.0767600
0	0.1468402

Explanation: (1 point) Expected they are not equal, because we do not have enough data to exactly solve for or estimate beta

```r
y_hat <- X %*% beta_hat
mean((y - y_hat)^2)
```

Compute MSE for predicting y (3 points)

```
## [1] 1.848893e-31
```

Explanation: (1 point) Expected to be numerically close to zero, because perfect prediction is possible when `p > n`.

```r
X_test <- matrix(rnorm(n * p), nrow = n)
y_test <- X_test %*% beta
y_test_hat <- X_test %*% X_pseudoinv %*% y
mean((y_test_hat - y_test)^2)
```

Generate a new sample of test data and compute the (in-distribution) test MSE (3 points)

```
## [1] 0.7142776
```

Explanation: (1 point) Expected that this would be larger than zero, because test error is generally higher than training error.

Use penalized regression to estimate beta (4 points) Compute the ridge and lasso estimates using `lambda = 0.1`, and compare these with the estimate from using `svd`. You may want to read the documentation for `?glmnet` and `?coef.glmnet`

```r
beta_ridge <- coef(glmnet(X, y, intercept = FALSE, alpha = 0, lambda = .1))
beta_lasso <- coef(glmnet(X, y, intercept = FALSE, lambda = .1))
```

```r
# Comparison
cbind(beta, beta_lasso[-1], beta_ridge[-1], beta_hat) |>
  kable()
```

beta			
1	0.7271206	0.2075285	0.2152325
0	0.0000000	-0.0000405	-0.1195434
0	0.0000000	0.4483238	-0.1159438
0	0.0000000	0.1004163	0.0664787
0	0.0000000	-0.0545385	-0.1588557
0	0.0000000	-0.1072776	-0.1190181
0	-0.1492376	-0.8173406	-0.2192990
0	0.0000000	0.0249320	0.1476748
0	0.0000000	-0.0941434	-0.0767600
0	0.0000000	0.0019428	0.1468402

Explanation: (1 point) Expected lasso solution to be sparse and ridge solution to not be sparse.

```r
y_test_hat <- X_test %*% beta_lasso[-1]
mean((y_test_hat - y_test)^2)
```

Compute test MSE using penalized regression estimates (3 points)

```
## [1] 0.05443351
```

```r
y_test_hat <- X_test %*% beta_ridge[-1]
mean((y_test_hat - y_test)^2)
```

```
## [1] 0.7803853
```

Explanation: (1 point) Expected or not, the lasso estimate is closest to the true beta and the test error is lower for the lasso compared to the other methods.