DS1 - Problem Set 2.

Son N. Nguyen

16 March 2022

Problem 1.

a.) From $min_b[\sum_{i=1}^{n}(Y_i - b)^2 + \lambda * b^2]$, $\lambda >= 0$ show that the solution is:

$$\hat{\beta}_0^{\,ridge} = \frac{\sum_{i=1}^{n} Y_i}{n + \lambda}$$

Step 1. Decompose square term and set up the partial derivative function w.r.t. b

$$min_b[\sum_{i=1}^{n}(Y_i - b)^2 + \lambda * b^2] = min_b[\sum_{i=1}^{n}(Y_i^2 - 2Y_i b + b^2) + \lambda * b^2]$$

Step 2. Take partial derivative of the target function

$$\frac{\partial}{\partial b}\sum_{i=1}^{n}(Y_i^2 - 2Y_i b + b^2) + \lambda * b^2 = \sum_{i=1}^{n}(-2Y_i + 2b) + 2\lambda * b$$

Step 3. Extract multipliers and term b from the sum formula and set it to zero according to FOC in optimization

$$-2\sum_{i=1}^{n} Y_i + 2bn + 2\lambda * b = 0$$

Step 4. Dividing the whole equation with 2 and factoring out b, we will arrive to:

$$-\sum_{i=1}^{n} Y_i + b(n + \lambda) = 0$$

Step 5. Moving the first term to the RHS of the equation and perform a division by $(n + \lambda)$ (it is not zero based on the assumption that n and lambda are two non-negative values), we will arrived to the desired solution, q.e.d.

$$\hat{\beta}_0^{\,ridge} = b = \frac{\sum_{i=1}^{n} Y_i}{(n + \lambda)}$$

Discussion: It's difference from the OLS coefficient estimator $\hat{\beta}_0^{\,OLS} = \overline{Y}$ is only the appearance of λ in the denominator (as $\frac{\sum_{i=1}^{n} Y_i}{n} = \overline{Y}$). Therefore, it means that a higher penalty parameter would lead to lower coefficient values, and intuitively would shrink more aggressively the estimated β coefficient towards zero (never reaches it actually).

b.) Suppose that $\beta_0 = 1$ and $\epsilon \sim N(0, \sigma^2)$ with $\sigma^2 = 4$. Generate a sample of size $n = 10$ from the model and compute $\hat{\beta}_0^{ridge}$ for a grid of values over the interval $[0, 20]$.

```
set.seed(1234)

n=10

#generate ys from the simplest model
generate_sample <- function(n) {
    e <- rnorm(n, mean = 0, sd = 2)
    1 + e
}

#implement derived formula for beta
compute_ridge <- function(lambda) {
  y <- generate_sample(n)
  rbeta <- sum(y)/(length(y)+lambda)
}

#simulation
run_simulation <- function(lambdas,n){

  map_df(lambdas, ~{
      model <- lapply(.x, compute_ridge)
      tibble(
          lambda = .x,
          beta_hat = as.numeric(model),
          beta_zero = 1,
          error = beta_zero - beta_hat
      )
  })

}

sim1 <- run_simulation(n=10, lambdas=seq(0,20, 0.5))

kable(x = head(sim1), digits = 3, caption = 'Head of Sample') %>%
  kable_styling(latex_options = c("hold_position","striped"), font_size = 8)
```

Table 1: Head of Sample

lambda	beta_hat	beta_zero	error
0.0	0.234	1	0.766
0.5	0.727	1	0.273
1.0	0.204	1	0.796
1.5	-0.463	1	1.463
2.0	-0.183	1	1.183
2.5	0.354	1	0.646

c.) Repeat part b), say, 1000 times so that you end up with 1000 estimates of β_0 for all the λ values that you have picked. For each value of λ, compute bias, variance, and MSE.

```
n = 10

# simulate 1k times
set.seed(1234)
nsim = 1000
simulation_results <- map_df(
    seq(nsim),
    run_simulation,
    lambdas = seq(0,20,1)
)

# add metrics
df <- group_by(simulation_results, lambda) |>
summarise(bias2 = mean(error)^2, var = var(beta_hat)) |>
mutate(MSE = bias2 + var)

kable(x = head(df), digits = 3, caption = 'Head of Error metrics') %>%
  kable_styling(latex_options = c("hold_position","striped"), font_size = 8)
```

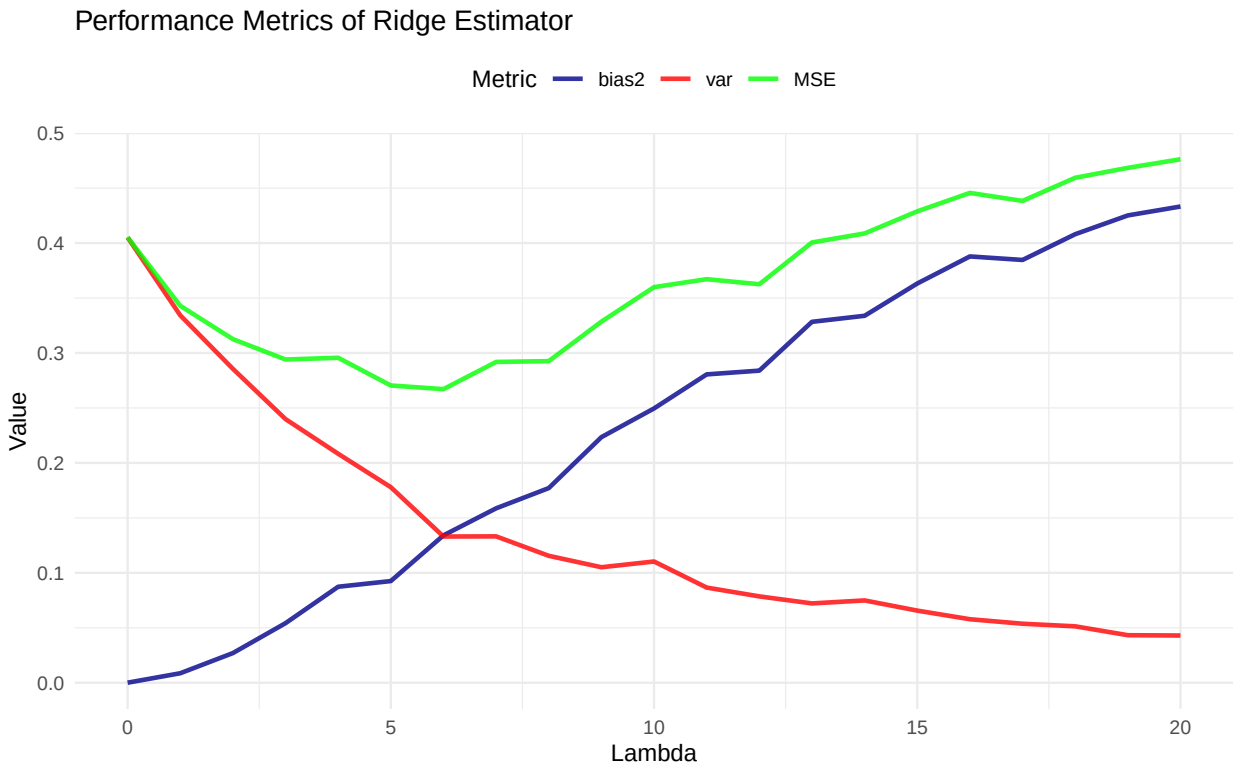
Table 2: Head of Error metrics

lambda	bias2	var	MSE
0	0.000	0.405	0.405
1	0.009	0.334	0.343
2	0.027	0.286	0.313
3	0.054	0.240	0.294
4	0.087	0.208	0.296
5	0.093	0.178	0.270

d.) Plot metrics as a function of λ and interpret the results.

```
visualize_simulation_results <- function(simulation_results) {
  pivot_longer(df, bias2:MSE, names_to = "metric") |>
  mutate(metric = factor(metric, levels = c("bias2", "var", "MSE"))) |>
  ggplot(aes(lambda, value, color = metric)) + geom_line(size = 1, alpha = 0.8) +
  labs(title = "Performance Metrics of Ridge Estimator",
     y = "Value",
     x = "Lambda") +
  scale_color_manual(name = "Metric", values = c("bias2" = "darkblue", "var" = "red", "MSE" = "green")) +
  theme(legend.position = 'top')
}

#visualize
visualize_simulation_results(df)
```



Discussion: As we shrink more aggressively out coefficients, **bias increase and variance decreases initially. This is a known pattern of shrinkage methods.** In overall, MSE decreases at lower lambdas because we managed to decreased variance with a larger pace than bias, gaining a bit of advantage (optimal at around lambda = 6), but **after the sweet spot, we are losing more accuracy in trade-off of variance and MSE starts to rise.**

Problem 2.

The population version of the optimization problem that defines the first principal component of the two variables is

$$max_{u1,u2} Var(u_1 X + u_2 Y) \; subject \; to \; u_1^2 + u_2^2 = 1$$

a.) Suppose that $Var(X) > Var(Y)$ and $cov(X,Y) = E(XY) = 0$. Derive the first principle component vector. Draw an illustrative picture and explain the result intuitively.

(Hint: expand the variance formula and substitute the constraint. Then carry out the minimization.)

Step 1. Use alternative formula of variance

$$max_{u1,u2}[u_1^2 Var(X) + u_2^2 Var(Y)] \; subject \; to \; u_1^2 + u_2^2 = 1$$

Step 2. Factor out u_1 from the variance term

$$max_{u1,u2}[\frac{u_1^2}{N}\sum_{i=1}^{n}(X_i - \mu_x)^2 + \frac{u_2^2}{N}\sum_{i=1}^{n}(Y_i - \mu_y)^2]$$

Step 3. Substitute the constraint and take full derivative of the optimization function

$$\frac{\partial}{\partial u_1}[\frac{u_1^2}{N}\sum_{i=1}^{n}(X_i-\mu_x)^2+\frac{1-u_1^2}{N}\sum_{i=1}^{n}(Y_i-\mu_y)^2]$$

Step 4. Set the derivative to 0 (FOC)

$$\frac{2u_1}{N}\sum_{i=1}^{n}(X_i-\mu_x)^2-\frac{2u_1}{N}\sum_{i=1}^{n}(Y_i-\mu_y)^2=0$$

Step 5. Simplify by dividing both sides with 1/N and 2

$$u_1\sum_{i=1}^{n}(X_i-\mu_x)^2-u_1\sum_{i=1}^{n}(Y_i-\mu_y)^2=0$$

Step 6. Factor out u_1 and plug in zero for both means as we know that they are zero for X and Y. In this case, either u_1 or the sum term has to be zero.

Note that deriving for u_2, we will arrive to the same end-result.

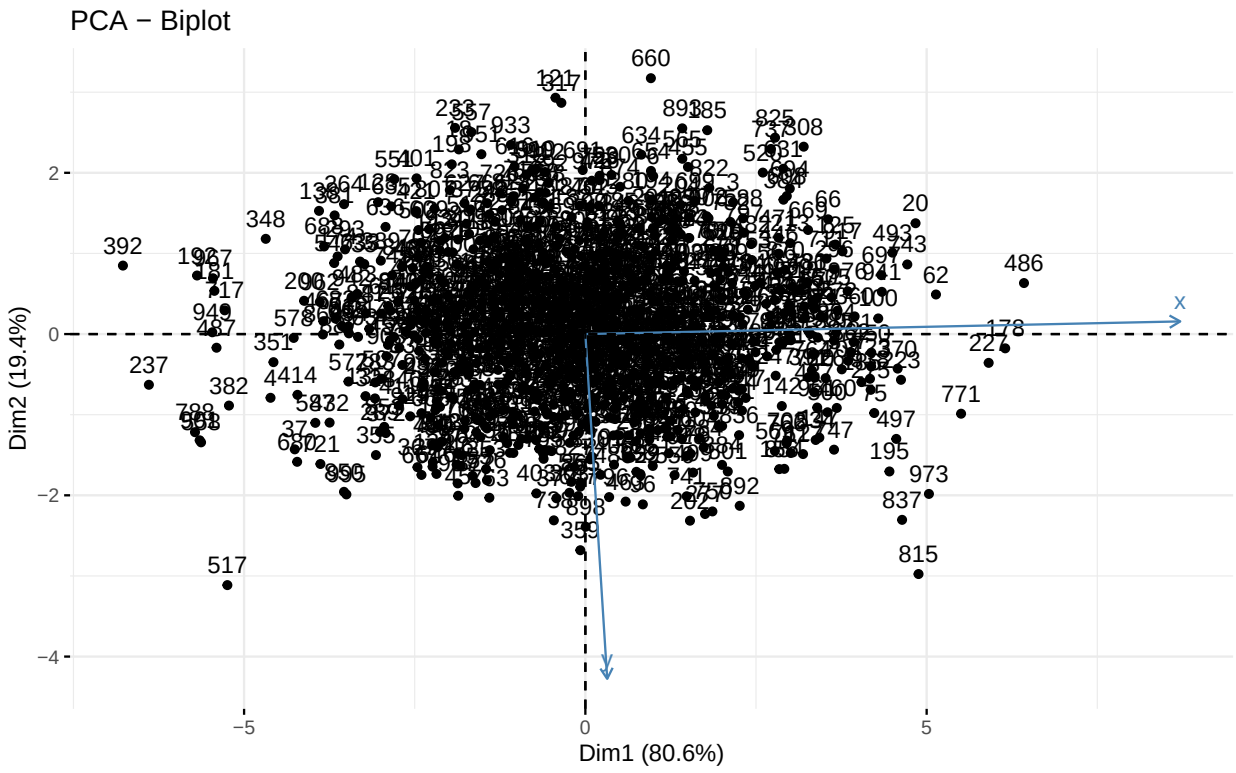
$$u_1(\sum_{i=1}^{n}[X_i^2-Y_i^2])=0$$

Step 7. Using the assumptions that $Var(X)>Var(Y)$ and $cov(X,Y)=E(XY)=0$, we can conclude that the above equation is only satisfied if $u_1^*=0$, given that X and Y are uncorrelated and the variance along the horizontal pane will be higher, thus we catch more variance with a vertical vector. This also means that $u_2^*=1$ from the constraint.

$$(u_1^*,u_2^*)=(0,1)$$

Discussion: We know that X and Y are uncorrelated and the first principal component vector can only take up values of (1,0) or (0,1). taking into account the other assumption that Var(X) > Var(Y) means that the horizontal dispersion is larger in overall. **All in all, the first principal vector will be** $(u_1^*,u_2^*)=(0,1)$**.** This is verified with the figure below which shows that larger percentage of variances can be explained along the X, thus the the first vector will pick up a vertical vector maximizing the horizontal variance. This is why PCA often uses scaling which evens out the playing field for both axis.

```
set.seed(1234)
x <- rnorm(1000, mean= 0, sd=2)
y <- rnorm(1000, mean= 0, sd=1)
pcdf <- cbind(x, y)
data_dep <- prcomp(pcdf, scale. = FALSE)
fviz_pca(data_dep)
```



b.) Suppose that $Var(X) = Var(Y) = 1$ (principle component analysis is often performed after standardization) and $cov(X, Y) = E(XY) = 0$. Show that in this case any vector (u_1, u_2) with length 1 is a principal component vector (i.e., it solves the problem above). Explain intuitively this puzzling result. (A picture can help.)

Start with the previously given optimization problem

$$max_{u1,u2}[u_1^2 Var(X) + u_2^2 Var(Y)] \; subject \; to \; u_1^2 + u_2^2 = 1$$

After scaling.... $Var(X) = Var(Y) = 1$

$$max_{u1,u2}[u_1^2 + u_2^2] \; subject \; to \; u_1^2 + u_2^2 = 1$$

Plugging int the constraint (length have to be unit re: Pythagoras theorem)

$$max_{u1}[u_1^2 + (1 - u_1^2)]$$

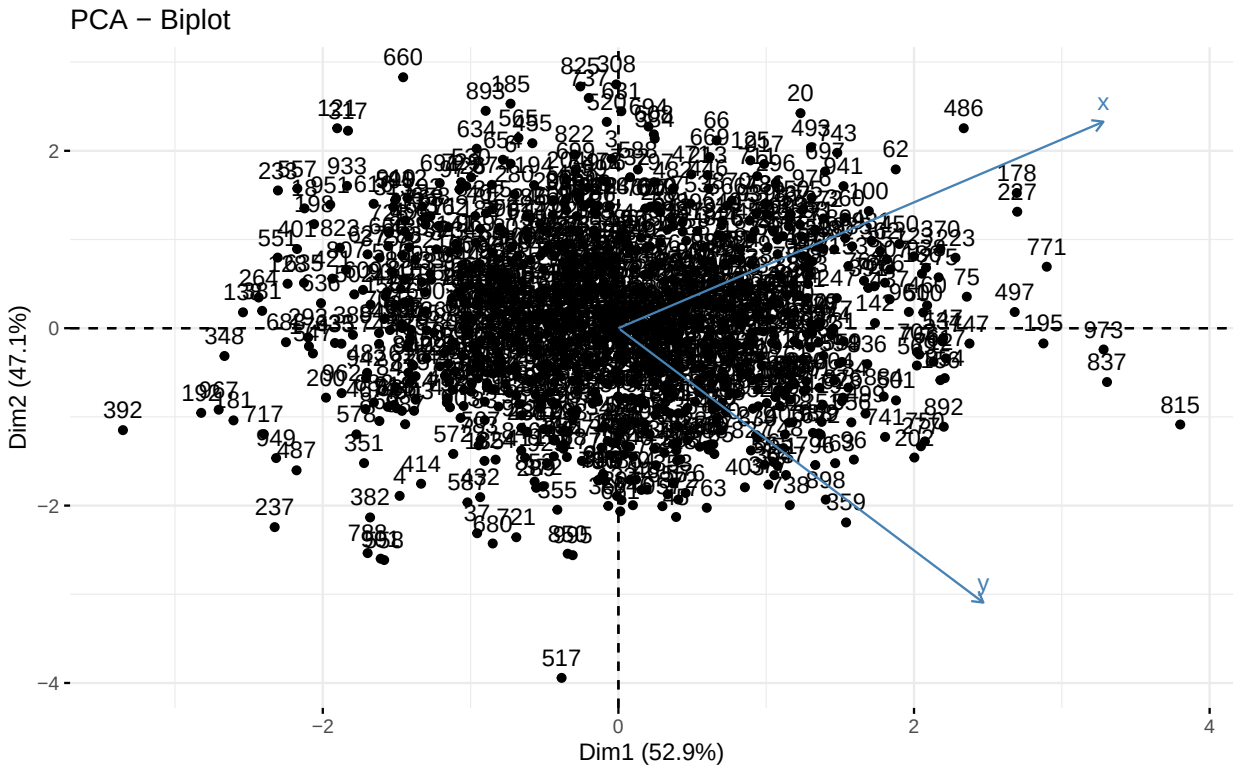
Subtracting the u_1 terms, we are left with a constant of 1.

$$max_{u1,u2}[1]$$

Discussion: It means that whatever u_1 and hence u_2 picks up w.r.t. the constrains that their sum of squares are equal to 1 (or length is 1), the variance is always going to be 1. This stems from the fact that the Variance of both X and Y is 1, thus **data points are scattering evenly around the origin**. We can set the vector in whatever direction along the unit circle, we will get the same variance fixed variance of 1 when projecting each observation to the line determined by the first principal component vector.

This is verified by the visualization below where I generated a sample of X, and Y with a unit variance (standardized). It is indicated in the graph that the **variance explained by each axis is around 50% which means that the principal component vector can pick up and arbitrary vector of length 1.**

```
set.seed(1234)
x <- rnorm(1000, mean= 0, sd=1)
y <- rnorm(1000, mean= 0, sd=1)
pcdf <- cbind(x, y)
data_dep <- prcomp(pcdf)
fviz_pca(data_dep)
```



Problem 3.

ISLR Exercise 3 in Section 6.8 (p. 260). Please use the version of the textbook posted online (7th printing).

Suppose we estimate the regression coefficients in a linear regression model by minimizing

$$RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 \ subject\ to \ \sum_{j=1}^{p}|\beta_j| \leq s$$

a.) As we increase s from 0, the training RSS will:

Answer: (iv) Steadily decrease – As we increase s from 0, all βs increase from 0 to their least square estimate values

b.) Repeat (a) for test RSS.

Answer: (ii) Decrease initially, and then eventually start increasing in a U shape: We start at the highest RSS when s=0 because than all the β coefficients are also zero. Same as in the training set, when we lift off from s=0, coefficients approach the OLS fit estimates and RSS starts to decrease. However, coefficients tend to overfit the sample training data which can increase the test RSS in the later stages.

c.) Repeat (a) for variance.

Answer: (iii) Steadily increase: When s=0, we have no variance whatsoever because our model estimates a constant (no coefficients picking up the variance of Xs). When we start to increase s from zero, βs start to appear and At this point, the values of s become highly dependent on training data, thus increasing the variance in overall.

d.) Repeat (a) for (squared) bias.

Answer: (iv) Steadily decrease: When s=0, bias is relatively large (estimate far off from the true value), since, we are estimating the constant. β coefficients start to increase once we push s to higher values, therefore decrease the bias gradually.

e.) Repeat (a) for the irreducible error.

Answer: (v) Remains constant: As the name states, this is an irreducible error, hence it is independent of how s evolves.