Math for Data Science: Problem Set 3

Sofia Breganni, David Moth, Adarsh Tripathi

2025-24-11

Due Date: Wednesday, December 3 by the end of the day (23:59).

Instructions: Please submit one solution set per group and include your group members' names at the top. Please write your solutions within this Rmd file, under the relevant question. Please submit the knitted output as a pdf. Make sure to show all code you used to arrive at the answer. However, please provide a brief, clear answer to every question rather than making us infer it from your output, and please avoid printing unnecessary output.

1. More Cat Compression

I have another cat named Lev. He was upset that Laszlo got to be in my course materials and he didn't. In this exercise we will explain to Lev why my initial choice of Laszlo was nothing personal.

Figure 1: My other cat, Lev.

a. Load the image of Lev. Extract its red, green, and blue matrices and store them as separate objects called `r`, `g`, and `b`. Center the blue matrix and compute its variance-covariance matrix. What is the dimensionality of this variance-covariance matrix and why?

```
library(jpeg)
library(patchwork)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
lev_the_cat <- readJPEG("lev.jpg")

r_lev <- lev_the_cat[,,1]
g_lev <- lev_the_cat[,,2]
b_lev <- lev_the_cat[,,3]

b_centered_lev <- scale(b_lev, center = TRUE, scale = FALSE)

vc_lev <- cov(b_centered_lev)

dim(vc_lev)
```

```
## [1] 200 200
```

The covariance matrix has dimensions 200×200 because we are capturing the covariance between pairs of features, not observations. The covariance matrix is defined by:

$$S = \frac{1}{n} X^\top X$$

In our case, the dimensions are:

$$X^\top \in \mathbb{R}^{200 \times 150}, \quad X \in \mathbb{R}^{150 \times 200} \Rightarrow X^\top X \in \mathbb{R}^{200 \times 200}$$

Thus, the product $X^\top X$ results in a 200×200 covariance matrix.

b. Find the total variance of the centered blue matrix for this image. Compare this to the total variance of the centered blue matrix for Laszlo.

```r
total_variance_blue_lev <- sum(diag(vc_lev))

laszlo_the_cat <- readJPEG("laszlo.jpg")

r_laszlo <- laszlo_the_cat[,,1]
g_laszlo <- laszlo_the_cat[,,2]
b_laszlo <- laszlo_the_cat[,,3]

b_centered_laszlo <- scale(b_laszlo, center = TRUE, scale = FALSE)

vc_laszlo <- cov(b_centered_laszlo)

total_variance_blue_laszlo <- sum(diag(vc_laszlo))
```

The total variance of blue for Lev the cat is 9.441, while for Laszlo (calculated using lab script), it is 13.1398.
I infer that the blue band in Laszlo's image carries more variability (greater contrast in pixel values) compared to Lev's image.

c. Use the `eigen` command to get the eigendecomposition of the blue variance-covariance matrix for Lev. Multiply the centered blue data matrix by the eigenvector corresponding to the largest eigenvalue to get the first principal component. Compute its variance.

```r
eigs_lev <- eigen(vc_lev)

first_eigenvector_lev <- eigs_lev$vectors[,1]

PC1_lev <- b_centered_lev %*% first_eigenvector_lev

lev_var_PC1 <- var(PC1_lev)
lev_var_PC1
```

```
##          [,1]
## [1,] 6.359234
```

d. Use the answers to the previous two questions to compute the proportion of variance explained for Lev's first principal component. Check your answer against a scree plot produced by the `fviz_eig` command.[1] Compare to what we saw for Laszlo in the lab.

```r
# Calculating Lev's

prop_explained_by_first_PC_lev <- lev_var_PC1 / total_variance_blue_lev

# Calculating Laszlo's
eigs_laszlo <- eigen(vc_laszlo)

first_eigenvector_laszlo <- eigs_laszlo$vectors[,1]

PC1_laszlo <- b_centered_laszlo %*% first_eigenvector_laszlo

laszlo_var_PC1 <- var(PC1_laszlo)

prop_explained_by_first_PC_laszlo <- laszlo_var_PC1 / total_variance_blue_laszlo

# Creating the Scree plots
b_pca_lev <- prcomp(b_lev, center = TRUE, scale. = FALSE)
b_pca_laszlo <- prcomp(b_laszlo, center = TRUE, scale. = FALSE)

p1 <- fviz_eig(b_pca_lev, barfill = "blue", ncp = 5, addlabels = TRUE) +
  ggtitle("Lev Blue PCA") +
  ylim(0, 100) +   # fixed y-axis
  theme(plot.title = element_text(hjust = 0.5))
```

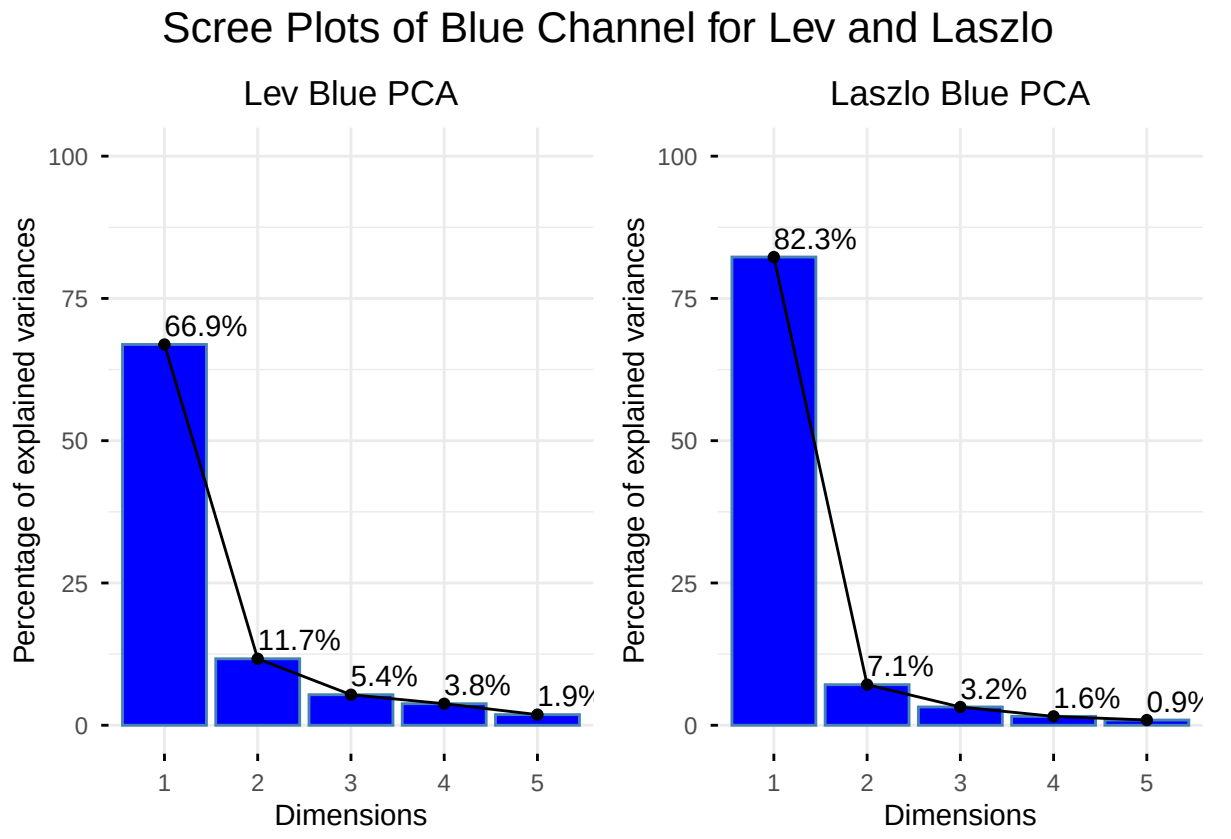
```
## Warning in geom_bar(stat = "identity", fill = barfill, color = barcolor, :
## Ignoring empty aesthetic: `width`.
```

```r
p2 <- fviz_eig(b_pca_laszlo, barfill = "blue", ncp = 5, addlabels = TRUE) +
  ggtitle("Laszlo Blue PCA") +
  ylim(0, 100) +   # same y-axis
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning in geom_bar(stat = "identity", fill = barfill, color = barcolor, :
## Ignoring empty aesthetic: `width`.
```

[1] If they are similar up to the second decimal place, that's good enough.

```r
(p1 + p2) + plot_annotation(
  title = "Scree Plots of Blue Channel for Lev and Laszlo",
  theme = theme(plot.title = element_text(hjust = 0.5, size = 16))
)
```



The proportion of variance explained by Lev's first principal component is 66.90%, while for Laszlo (calculated using the lab script) it was 82.3%.
This indicates that Laszlo's first principal component in the blue band captures more variability (i.e., more information) compared to that of Lev.

e. Now compute Lev's second principal component. Compute the covariance of the first principal component with the second.[2] Is this what you expected? Comment briefly.

```r
second_eigenvector_lev <- eigs_lev$vectors[,2]

PC2_lev <- b_centered_lev %*% second_eigenvector_lev

first_second_covariance_lev <- cov(as.vector(PC1_lev), as.vector(PC2_lev))
first_second_covariance_lev <- round(first_second_covariance_lev, 10)
```

[2] You can round to 10 decimal places.

The covariance of the two first principal components is 0. This makes sense, as principal components by design are orthogonal, so that the maximum amount of unique and "different" variance can be captured, as otherwise the data that is captured can be explained by the former principal component.

f. Now let's run PCA on Lev's `r`, `g`, and `b` matrices using the `prcomp` function with the options `center=FALSE` and `scale.=FALSE`.[3] Combine these objects into a list. Now, looping over a handful of numbers of principal components, reconstitute images of Lev as we did in your lab. How many principal components does it take to start to recognize Lev as a cat?

```
r_pca <- prcomp(r_lev, center = FALSE, scale. = FALSE)
g_pca <- prcomp(g_lev, center = FALSE, scale. = FALSE)
b_pca <- prcomp(b_lev, center = FALSE, scale. = FALSE)
rgb_pca_lev <- list(r_pca, g_pca, b_pca)

vec <- c(1, 2, 5, 10, 12, 15, 20, 50, 100)

h <- nrow(r_lev)
w <- ncol(r_lev)

par(mfrow=c(3,3), mar=c(0.5,0.5,2,0.5))

for(i in vec){
  photo_i <- sapply(rgb_pca_lev, function(j) {
    new.RGB <- j$x[, 1:i] %*% t(j$rotation[, 1:i])
    (new.RGB - min(new.RGB)) / (max(new.RGB) - min(new.RGB))
  }, simplify = "array")

  plot(1, type="n", xlim=c(0, w), ylim=c(0, h), asp=1, axes=FALSE, xlab="", ylab="")

  rasterImage(photo_i, 0, 0, w, h)

  mtext(paste(i, "PCs"), side=3, line=0.5, cex=1.2)
}
```

[3] Scaling and centering is desirable when your variables are on different scales, but in this case it messes up the colors.

As can be seen above, it takes around 10 principal components to have a reasonable degree of confidence that we are indeed looking at a cat.

g. Using what you learned from parts (b), (d), and (f) above, explain why Lev requires more principal components to be minimally represented than Laszlo, despite being equally beautiful.

Lev needs more principal components to be represented because his image spreads the variance across many directions. In part (b) we saw his blue band had lower total variance than Laszlo's, and in part (d) his first PC only explained about 66.9% compared to Laszlo's 87.6%. So when we tried to reconstruct (part f), Lev only looked like a cat after around 10–20 PCs, while Laszlo was recognizable with fewer. Basically, Laszlo's image is easier to compress since most of the information is in the first component, but Lev's takes more PCs to capture the details.
And, despite all this math, both Lev and Laszlo are equally beautiful cats; PCA just tells us about pixel variance, not about their actual charm.

2. Penalized Regression

We will derive the estimator for ridge regression, which is one of several *penalized regression* methods.[4] The process we will follow is very similar to the standard regression estimator we derived in your lab, but with a

[4]See p. 237-244 of ISL.

small change to the loss function. Rather than minimizing the sum of squared errors, we will minimize the sum of squared errors subject to a constraint:

$$||\beta||_2^2 \leq s$$

where s is just some constant chosen by the analyst. Recall that $||\beta||_2^2$ is the squared L_2 norm of the β vector.[5] This question will build on the concepts and data in the lab, so please revisit those materials if anything here is unclear.

a. Write down the Lagrangian for this constrained minimization problem. Please use matrix notation (including for the L_2 norm).

We want to minimize the sum of squared residuals:

- $\min_{\beta} \; ||y - \mathbf{X}\beta||_2^2$
- Where $||y - \mathbf{X}\beta||_2^2 = \sum_{i=1}^{n}(y_i - (\mathbf{X}\beta)_i)^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$

With an L_2 norm constraint:

- The L_2^2 norm will be: $||\beta||_2^2 = \sum_{i=1}^{p} \beta^2 = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{\beta}$
- And the constraint function: $g(\beta) = \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{\beta} - s$

$$\mathcal{L}(\boldsymbol{\beta}, \lambda) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda(\boldsymbol{\beta}^T\boldsymbol{\beta} - s)$$

b. Take the first derivative of the Lagrangian with respect to β and set it equal to $\mathbf{0}$ to get the first order condition. (If you're feeling retro, you can use the Matrix Cookbook to help you.) Solve for $\hat{\beta}$.

[5] Also known as the Euclidian norm.

Solving the derivative step-by-step:

- Derivative of $(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\mathbf{X}^T X\boldsymbol{\beta}$

$$\frac{d}{d\beta}(\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}) = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T X\boldsymbol{\beta}$$

- Derivative of $\lambda(\boldsymbol{\beta}^T\boldsymbol{\beta} - s)$

$$\frac{d}{d\beta}\lambda(\boldsymbol{\beta}^T\boldsymbol{\beta} - s) = \lambda 2\boldsymbol{\beta}$$

- Therefore:

$$\frac{d\mathcal{L}(\boldsymbol{\beta}, \lambda)}{d\beta} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda 2\boldsymbol{\beta}$$

Setting the derivative $= 0$ and solving for $\hat{\beta}$:

$$\begin{aligned} -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda 2\boldsymbol{\beta} &= \mathbf{0} \\ 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \lambda 2\boldsymbol{\beta} &= 2\mathbf{X}^T\mathbf{y} \\ (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\boldsymbol{\beta} &= \mathbf{X}^T\mathbf{y} \\ \hat{\boldsymbol{\beta}} &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{y}) \end{aligned}$$

c. Compare your answer to the standard linear regression estimator. Under what condition are they the same?

From the lectures we found that:

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

From (b) we know that:

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

The ridge estimator modifies the OLS estimator by adding a the penalty term $\lambda\mathbf{I}$ on the size of the coefficients. When $\lambda = 0$, the penalty disappears and the ridge estimator reduces exactly to the OLS estimator. Therefore, we can say that OLS is a special case of ridge regression with no penalization.

d. Using the same `BostonHousing` dataset and the same set of variables from the lab, compute the ridge regression $\hat{\beta}$ with the equation you found in part (b) above. First standardize your $\mathbf{X}$ matrix using the `scale()` function in `R`. Use the `glmnet()` function in the `glmnet` package to check your answer.[6]

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

```
library(Matrix)
library(glmnet)
```

```
## Loaded glmnet 4.1-10
```

[6]Note that `glmnet`'s `lambda` parameter corresponds to your $\frac{\lambda}{N}$, where N is the number of rows of your data. Also, your estimates may differ slightly from `glmnet`'s, at around the second decimal place. That's alright; this is likely due to `glmnet`'s optimization algorithm. It's computationally expensive to invert large matrices so `glmnet` is probably taking some more efficient but (slightly) less precise approach.

```r
library(mlbench)
data(BostonHousing)

# Design matrix
X_raw <- as.matrix(BostonHousing[,c("crim", "chas", "nox", "dis", "ptratio", "rad")],
                   ncol = 6,
                   byrow = TRUE)
X_raw <- apply(X_raw, 2, as.numeric)

X_stan <- scale(X_raw)
X <- cbind(1, X_stan)

# Response vector
y <- BostonHousing$medv

# Finding Beta
lambda <- 1
I <- diag(ncol(X))

beta <- solve(t(X) %*% X + lambda*I) %*% (t(X) %*% y)
beta
```

```
##               [,1]
##         22.488363
## crim    -2.117056
## chas     1.211308
## nox     -5.608049
## dis     -2.763344
## ptratio -4.372866
## rad      1.914031
```

```r
fit <- glmnet(X, y, alpha = 0, lambda = lambda/nrow(X))
coef(fit)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 22.532806
##              .
## crim        -2.127686
## chas         1.211766
## nox         -5.651956
## dis         -2.796078
## ptratio     -4.389615
## rad          1.942497
```

e. For the sequence of lambdas below, make a plot with `lambda_seq` on the x-axis (increasing from 0 to 100) and the estimated β coefficients for per capita crime rate in the town (in blue), proximity to the Charles River (in red), and nitric oxides concentration (in green) on the y-axis. Use `geom_line` to plot the coefficients and add a black horizontal line at $y = 0$. Label your axes and include a legend.

variables:

- `crim`: per capita crime rate in the town (BLUE)

- `chas`: Charles River dummy variable (1 if tract is on the water, 0 if not) (RED)
- `nox`: nitric oxides concentration (parts per 10 million) (GREEN)

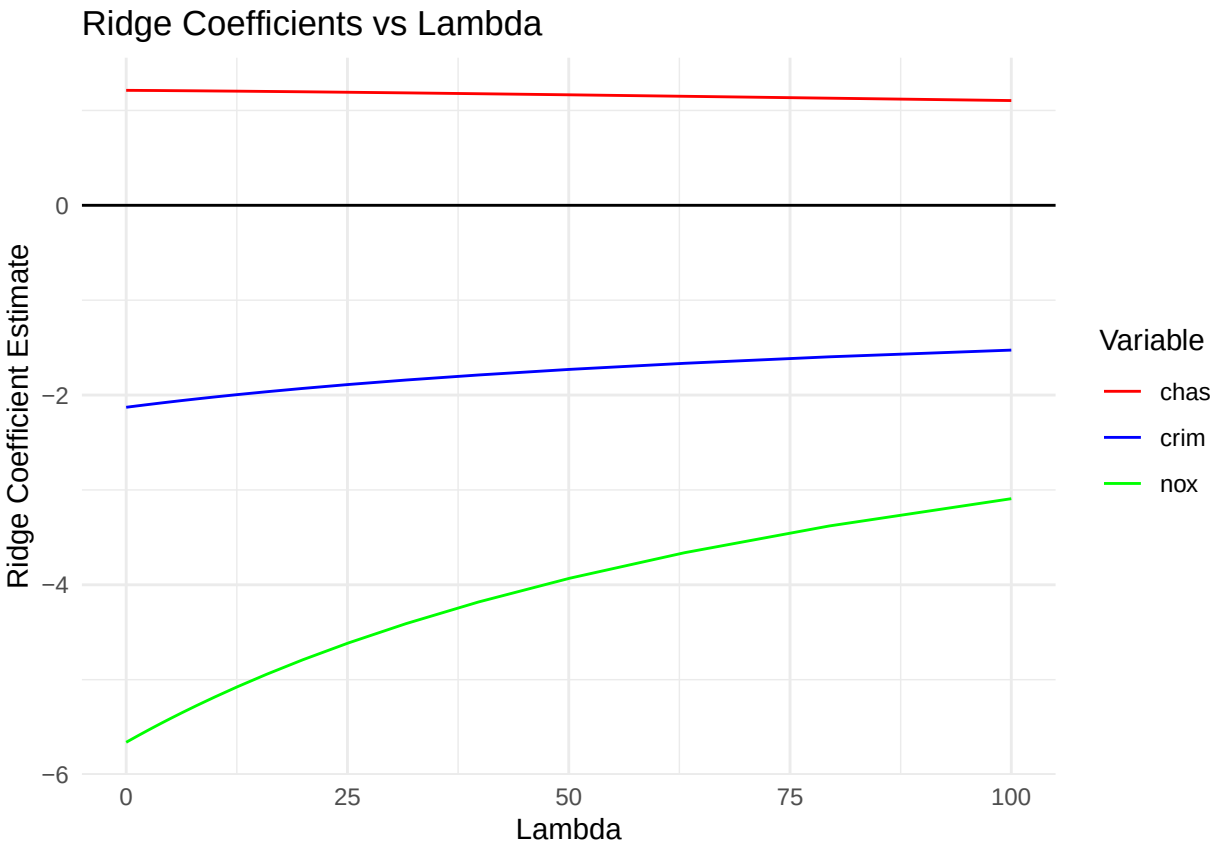
```
lambda_seq <- c(10^seq(2, -1, by = -.1), 0)

beta_plot <- sapply(lambda_seq, function(l){solve(t(X) %*% X + l*I) %*% (t(X) %*% y)})
beta_df <- as.data.frame(t(beta_plot))
colnames(beta_df) <- c("", "crim", "chas", "nox", "dis", "ptratio", "rad")
beta_df$lambda <- lambda_seq

beta_long <- beta_df[, c("crim", "chas", "nox", "lambda")] |>
  tidyr::pivot_longer(
    cols = -lambda,
    names_to = "variable",
    values_to = "coefficient"
  )

library(ggplot2)

ggplot(beta_long, aes(x = lambda, y = coefficient, color = variable)) +
  geom_line() +
  geom_hline(yintercept = 0, color = "black", linewidth = 0.5) +
  scale_color_manual(values = c(crim = "blue",
                                chas = "red",
                                nox = "green")) +
  labs(x = "Lambda",
       y = "Ridge Coefficient Estimate",
       color = "Variable",
       title = "Ridge Coefficients vs Lambda") +
  theme_minimal()
```



f. Discussion:

- Based on your plot above, explain why ridge is one of a number of so-called "shrinkage" estimators.
- Tie this back to the constrained optimization problem you solved to obtain the ridge regression $\hat{\beta}$. Can you see how a larger value of λ (or equivalently a small value of s) corresponds to greater shrinkage?
- You will learn more about this class of estimators and their virtues next semester, but do you have any intuitions as to when and why they might be desirable?

From the graph we can observe how, as λ grows, the coefficient estimates become closer and closer to 0. This is why ridge belongs to the family of "shrinkage" estimators: the penalty term forces the estimated coefficients to shrink in magnitude. The larger the value of λ, the stronger this pull toward zero, and thus the greater the shrinkage.
This shrinking behaviour connects back to the constrained optimization problem: in the constrained form, ridge regression minimizes the sum of squared residuals subject to the requirement that the coefficients lie within an L_2 ball:

$$\|\beta\|_2^2 \leq s.$$

A small value of s means the feasible region for β is tight; when s is small, the optimizer is not allowed to choose a large coefficient vector even if doing so would achieve a smaller residual sum of squares. In the Lagrangian form, this corresponds to a large value of λ. Thus, *large λ (equivalently, small s) imposes a strong restriction on the overall size of the coefficient vector*, which forces the solution to move closer to zero.
Ridge shrinkage estimator can be suitable when the predictors are supected to be highly correlated so it may bring multicollinearity OLS assumption violation in the model estimation. This is more visible as dimensions are increased. Ridge estimator introduces a small bias(through penalising term) to reduce variance compared to OLS. This helps coefficient to shrink but never to zero because it is assumed that all estimators contribute to the model.