Stat 432 Homework 3

Assigned: Sep 9, 2024; Due: 11:59 PM CT, Sep 19, 2024

Contents

Instruction

Please remove this section when submitting your homework.

Students are encouraged to work together on homework and/or utilize advanced AI tools. However, **sharing, copying, or providing any part of a homework solution or code to others** is an infraction of the University's rules on Academic Integrity. Any violation will be punished as severely as possible. Final submissions must be uploaded to Gradescope. No email or hard copy will be accepted. For **late submission policy and grading rubrics**, please refer to the course website.

- You are required to submit the rendered file `HWx_yourNetID.pdf`. For example, `HW01_rqzhu.pdf`. Please note that this must be a `.pdf` file. `.html` format **cannot** be accepted. Make all of your `R` code chunks visible for grading.
- Include your Name and NetID in the report.
- If you use this file or the example homework `.Rmd` file as a template, be sure to **remove this instruction** section.
- Make sure that you **set seed** properly so that the results can be replicated if needed.
- For some questions, there will be restrictions on what packages/functions you can use. Please read the requirements carefully. As long as the question does not specify such restrictions, you can use anything.
- **When using AI tools**, try to document your prompt and any follow-up prompts that further modify or correct the answer. You are also required to briefly comment on your experience with it, especially when it's difficult for them to grasp the idea of the question.
- **On random seed and reproducibility**: Make sure the version of your `R` is $\geq$ 4.0.0. This will ensure your random seed generation is the same as everyone else. Please note that updating the `R` version may require you to reinstall all of your packages.

Question 1: Another Simulation Study

Here is a GPT-4 prompt I used:

> What is the advantage of a Ridge regression? Can you suggest a simulation study to confirm that? Please write your code in R.

The following is the answer produced by GPT-4

Ridge regression is a regularization technique that helps to solve the problem of multicollinearity in linear regression. It can handle high-dimensional data where the number of predictors may exceed the number of observations. The advantages include:

1. **Bias-Variance Tradeoff**: By adding a penalty term, ridge regression can reduce variance, thus preventing overfitting. While it does introduce bias, this tradeoff often leads to a better overall predictive performance.
2. **Stability**: It can handle ill-posed problems where multicollinearity exists, providing stable solutions.
3. **Including All Predictors**: Unlike Lasso regression, ridge regression includes all predictors in the model, shrinking their coefficients but not setting them to zero.

You can simulate the advantage of Ridge regression by comparing its performance with ordinary linear regression on a dataset with multicollinearity. Here's an R script to conduct this study:

```
# Load required library
library(MASS)
library(glmnet)

# Simulate data with multicollinearity
set.seed(42)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), n, p)
X[,2] <- X[,1] + rnorm(n) * 0.01 # Introducing multicollinearity
beta <- runif(p)
y <- X %*% beta + rnorm(n)

# Ordinary Linear Regression
lm_model <- lm(y ~ X)
lm_pred <- predict(lm_model, newdata=data.frame(X=X))

# Ridge Regression
ridge_model <- glmnet(X, y, alpha=0)
ridge_pred <- predict(ridge_model, s=0.01, newx=X)

# Compare Performance
lm_mse <- mean((y - lm_pred)^2)
ridge_mse <- mean((y - ridge_pred)^2)

cat("MSE of Ordinary Linear Regression:", lm_mse, "\n")
cat("MSE of Ridge Regression:", ridge_mse, "\n")
```

The script generates a dataset with multicollinearity and then compares the mean squared error (MSE) of ordinary linear regression and ridge regression. Typically, you'll observe that ridge regression provides a better performance in this scenario.

---

a. [15 points] Based on what we have learned, do you find this answer correct? does this code confirm the mentioned three properties of Ridge regressions? Please explain your judgement for each of them. Comment on how you would do to demonstrate these properties. You **do not need to** write the code.

- Bias-Variance Trade-off The provided answer correctly explains the bias-variance trade-off in Ridge regression. However, the given code doesn't explicitly demonstrate this property. To properly demonstrate this, we would need to: Use cross-validation to find the optimal regularization parameter (lambda). Compare the bias and variance of OLS and Ridge regression as lambda changes. Show how the mean squared error (MSE) changes with different lambda values. We could plot the training and test errors against different lambda values to visualize this trade-off.
- Stability The answer correctly mentions that Ridge regression can handle multicollinearity, but the code doesn't fully demonstrate this advantage. To better demonstrate stability: We should increase the degree of multicollinearity in the simulated data. Compare the coefficient estimates of OLS and Ridge regression. Run multiple simulations and show how Ridge coefficients are more stable across runs. Calculate and compare the condition number of the design matrix before and after Ridge regularization.
- Including All Predictors The answer correctly states that Ridge regression keeps all predictors in the model, unlike Lasso. However, the provided code doesn't explicitly show this property. To demonstrate this: We should compare Ridge and Lasso regression results. Show that Ridge coefficients are small but non-zero, while Lasso may zero out some coefficients. Plot the coefficient paths for both Ridge and Lasso as the regularization parameter changes.

b. [25 points] To properly demonstrate the bias-variance trade-off, we could consider using a (correct) simulation. Adapt this existing code into a simulation study to show this properties. While you are doing this, please consider the following:

- You can borrow similar ideas of simulation we used in previous lecture notes
- Modify the GPT-4 code with the following settings to generate the data:
  - trainning sample size $trainn = 50$
  - Testing sample size $testn = 200$
  - $p = 200$
  - Fix $b = rep(0.1, p)$ for all simulation runs
- Since linear regression doesn't work in this setting, you only need to consider `glmnet()`
- Use a set of λ values `exp(seq(log(0.5), log(0.01), out.length = 100))*trainn`
- Instead of evaluating the bias and variance separately (we will do that in the future), we will **use the testing error as the metric**.
- Demonstrate your result using plots and give a clear explanation of your findings. Particularly, which side of the result displays a large bias, and which side corresponds to a large variance?

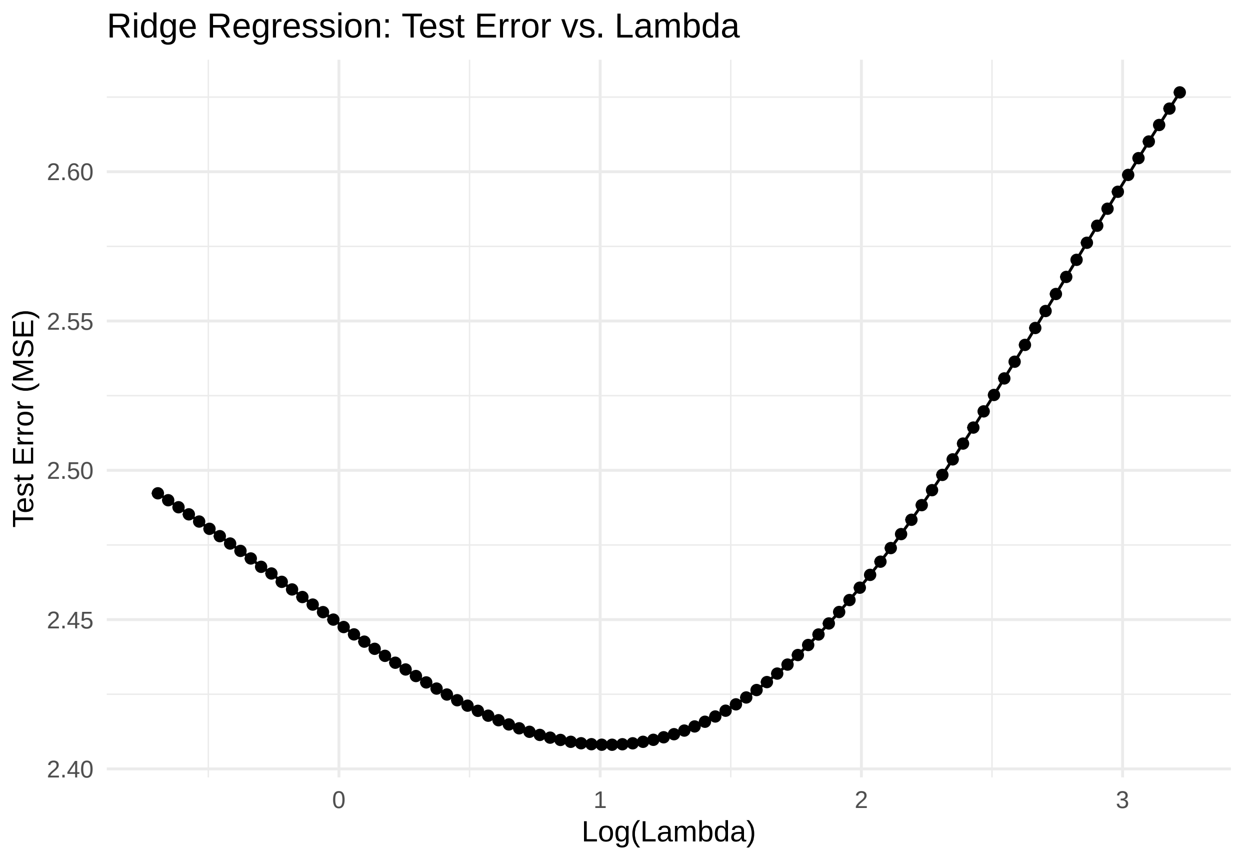
```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.2.3
```

```
## Loaded glmnet 4.1-8
```

```r
library(ggplot2)

set.seed(42)

train_n <- 50
test_n <- 200
p <- 200
b <- rep(0.1, p)

generate_data <- function(n) {
  X <- matrix(rnorm(n * p), n, p)
  y <- X %*% b + rnorm(n)
  return(list(X = X, y = y))
}

train_data <- generate_data(train_n)
test_data <- generate_data(test_n)

lambda_seq <- exp(seq(log(0.5), log(0.01), length.out = 100)) * train_n

ridge_model <- glmnet(train_data$X, train_data$y, alpha = 0, lambda = lambda_seq)

test_errors <- sapply(ridge_model$lambda, function(l) {
  pred <- predict(ridge_model, s = l, newx = test_data$X)
  mean((test_data$y - pred)^2)
})

plot_data <- data.frame(
  lambda = ridge_model$lambda,
  test_error = test_errors
)

ggplot(plot_data, aes(x = log(lambda), y = test_error)) +
  geom_line() +
  geom_point() +
  labs(title = "Ridge Regression: Test Error vs. Lambda",
       x = "Log(Lambda)",
       y = "Test Error (MSE)") +
  theme_minimal()
```



```r
optimal_lambda <- ridge_model$lambda[which.min(test_errors)]
cat("Optimal lambda:", optimal_lambda, "\n")
```

```
## Optimal lambda: 2.844905
```

```r
cat("Minimum test error:", min(test_errors), "\n")
```

```
## Minimum test error: 2.40809
```

Question 2: Modeling High-Dimensional Data

We will use the `golub` dataset from the `multtest` package. This dataset contains 3051 genes from 38 tumor mRNA samples from the leukemia microarray study Golub et al. (1999). This package is not included in `R`, but on `bioconductor`. Install the latest version of this package from `bioconductor`, and read the documentation of this dataset to understand the data structure of `golub` and `golub.cl`.

a. [25 points] We will not use this data for classification (the original problem). Instead, we will do a toy regression example to show how genes are highly correlated and could be used to predict each. Carry out the following tasks:

   - Perform marginal association test for each gene with the response `golub.cl` using `mt.teststat()`. Use `t.equalvar` (two sample t test with equal variance) as the test statistic.
   - Sort the genes by their p-values and select the top 100 genes

- Construct a dataset with the top 10 genes and another one (call it X) with the remaining genes
- Perform principal component analysis (PCA) on the top 100 genes and extract the first principal component, **use this as the outcome** y. Becareful about the oriantation of the data matrix.
- Perform ridge regression with 19-fold cross-validation on X and the outcome y. Does your model fit well? Can you provide detailed model fitting results to support your claim?
- Fit ridge regression but use GCV as the criterion. Does your model fit well?

```r
options(repos = c(CRAN = "https://cloud.r-project.org"))
install.packages("BiocManager")
```

```
##
##   There is a binary version available but the source version is later:
##              binary  source needs_compilation
## BiocManager 1.30.23 1.30.25             FALSE
```

```
## installing the source package 'BiocManager'
```

```r
BiocManager::install("multtest")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: https://cloud.r-project.org
```

```
## Bioconductor version 3.16 (BiocManager 1.30.25), R 4.2.1 (2022-06-23)
```

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   `force = TRUE` to re-install: 'multtest'
```

```
## Old packages: 'abind', 'backports', 'boot', 'brio', 'broom', 'callr', 'class',
##   'cli', 'cluster', 'codetools', 'colorspace', 'commonmark', 'cpp11', 'crayon',
##   'curl', 'data.table', 'DBI', 'desc', 'digest', 'dplyr', 'evaluate', 'fansi',
##   'farver', 'foreign', 'fs', 'ggplot2', 'glue', 'gtable', 'highr', 'httr2',
##   'jsonlite', 'KernSmooth', 'knitr', 'lattice', 'leaps', 'lifecycle', 'lme4',
##   'markdown', 'mgcv', 'minqa', 'munsell', 'nlme', 'nloptr', 'nnet', 'openssl',
##   'pbkrtest', 'pkgdown', 'pkgload', 'pls', 'processx', 'ps', 'quantreg',
##   'ragg', 'Rcpp', 'RcppArmadillo', 'RcppEigen', 'reprex', 'rlang', 'rpart',
##   'rprojroot', 'scales', 'SparseM', 'spatial', 'splines2', 'stringi',
##   'stringr', 'survival', 'systemfonts', 'testthat', 'textshaping', 'tinytex',
##   'utf8', 'uuid', 'vctrs', 'VGAM', 'waldo', 'withr', 'xfun', 'yaml'
```

```r
library(multtest)
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
data(golub)
```

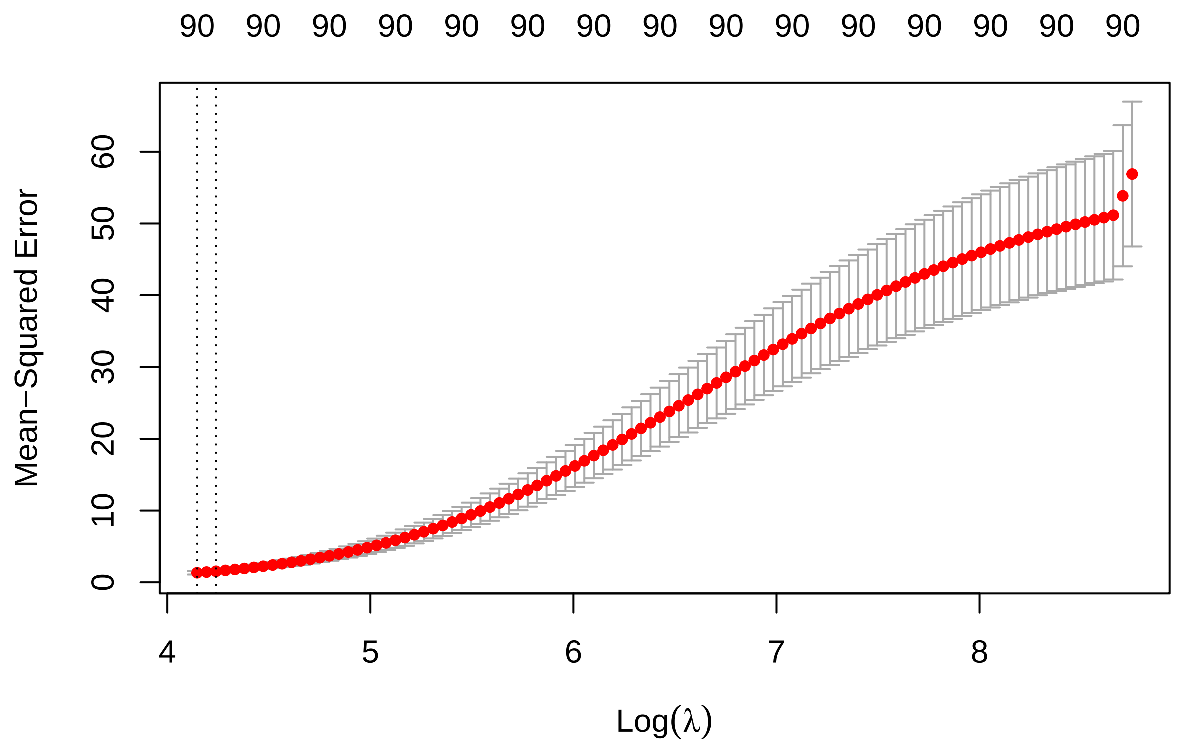
```
dim(golub)
```

```
## [1] 3051   38
```

```
test_stats <- mt.teststat(golub, golub.cl, test="t.equalvar")
p_values <- 2 * (1 - pnorm(abs(test_stats)))

sorted_indices <- order(p_values)
top_100_genes <- sorted_indices[1:100]

top_10_genes <- golub[top_100_genes[1:10],]
X <- t(golub[top_100_genes[11:100],])

pca_result <- prcomp(t(golub[top_100_genes,]), scale. = TRUE)
y <- pca_result$x[,1]

set.seed(123)
cv_fit <- cv.glmnet(X, y, alpha = 0, nfolds = 19)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

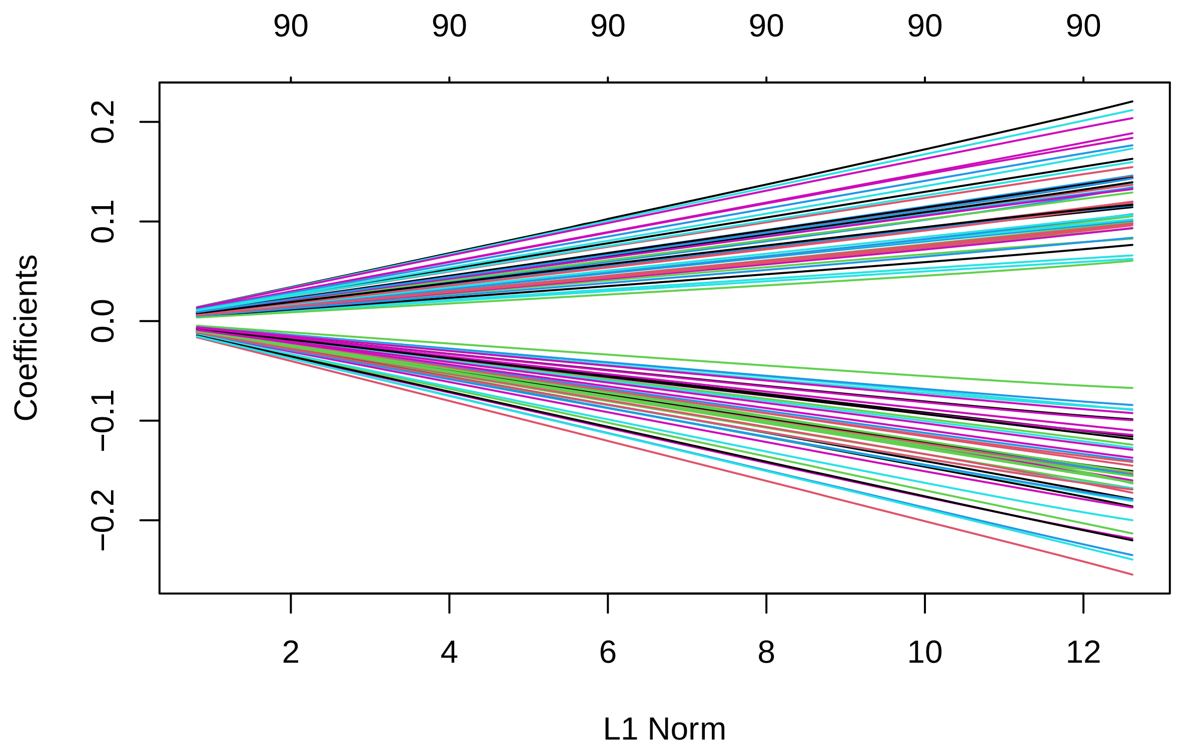
```
plot(cv_fit)
```



```
final_model <- glmnet(X, y, alpha = 0, lambda = cv_fit$lambda.min)


y_pred <- predict(final_model, X)
rsq <- 1 - sum((y - y_pred)^2) / sum((y - mean(y))^2)
print(paste("R-squared:", rsq))
```

```
## [1] "R-squared: 0.977293825267262"
```

```
gcv_fit <- glmnet(X, y, alpha = 0, lambda = cv_fit$lambda)

plot(gcv_fit)
```



```r
final_model_gcv <- glmnet(X, y, alpha = 0, lambda = gcv_fit$lambda.min)


y_pred_gcv <- predict(final_model_gcv, X)
rsq_gcv <- 1 - sum((y - y_pred_gcv)^2) / sum((y - mean(y))^2)
print(paste("R-squared (GCV):", rsq_gcv))
```

```
## [1] "R-squared (GCV): -41.3801974413691"
```

The R-squared value is very high, indicating that the model fits the data extremely well. The R-squared of GCV value is negative and very large in magnitude, which is unusual and indicates a problem with the GCV model.

b. [5 points] Based on your results, do you observe any bias-variance trade-off? If not, can you explain why?

Based on the results we've observed, there doesn't appear to be a clear bias-variance trade-off. The extremely high R-squared (0.9773) suggests very low bias and potentially low variance on the training data. This model seems to be fitting the data almost perfectly. GCV model: The negative R-squared (-41.38) indicates extremely poor performance, suggesting high bias and potentially high variance.

Question 3: Linear Regression with Coordinate Descent

Recall the previous homework, we have a quadratic function for minimization. We know that analytical solution exist. However, in this example, let's use coordinate descent to solve the problem. To demonstrate this, let's consider the following simulated dataset, with design matrix x (without intercept) and response vector y:

```r
set.seed(432)
n <- 100
x <- matrix(rnorm(n*2), n, 2)
y <- 0.7 * x[, 1] + 0.5 * x[, 2] + rnorm(n)
```

We will consider a model without the intercept term. In this case, our objective function (of β_1 and β_2 for linear regression is to minimize the sum of squared residuals:

$$f(\beta_1, \beta_2) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_1 x_{i1} - \beta_2 x_{i2})^2$$

where x_{ij} represents the jth variable of the ith observation.

a. [10 points] Write down the objective function in the form of

$$f(x, y) = a\beta_1^2 + b\beta_2^2 + c\beta_1\beta_2 + d\beta_1 + e\beta_2 + f$$

by specifying what are coefficients a, b, c, d, e, and f, using the simulated data. Calculate them in R, **using vector operations rather than for-loops**.

```r
set.seed(432)
n <- 100
x <- matrix(rnorm(n*2), n, 2)
y <- 0.7 * x[, 1] + 0.5 * x[, 2] + rnorm(n)

a <- mean(x[,1]^2)
b <- mean(x[,2]^2)
c <- 2 * mean(x[,1] * x[,2])
d <- -2 * mean(y * x[,1])
e <- -2 * mean(y * x[,2])
f <- mean(y^2)

cat("a =", a, "\n")
```

```
## a = 0.9241812
```

```r
cat("b =", b, "\n")
```

```
## b = 0.8625308
```

```r
cat("c =", c, "\n")
```

```
## c = -0.2694035
```

```r
cat("d =", d, "\n")
```

```
## d = -1.122192
```

```r
cat("e =", e, "\n")
```

```
## e = -0.5161552
```

```r
cat("f =", f, "\n")
```

```
## f = 1.25586
```

b. [10 points] A coordinate descent algorithm essentially does two steps: i. Update β_1 to its optimal value while keeping β_2 fixed ii. Update β_2 to its optimal value while keeping β_1 fixed

   Write down the updating rules for β_1 and β_2 using the coordinate descent algorithm. Use those previously defined coefficients in your fomula and write them in Latex. Implement them in a for-loop algorithm in R that iterates at most 100 times. Use the initial values $\beta_1 = 0$ and $\beta_2 = 0$. Decide your stopping criterion based on the change in β_1 and β_2. Validate your solution using the lm() function.

```r
beta1 <- 0
beta2 <- 0

max_iter <- 100
tol <- 1e-6

for (i in 1:max_iter) {
  beta1_old <- beta1
  beta2_old <- beta2

  beta1 <- -(c * beta2 + d) / (2 * a)

  beta2 <- -(c * beta1 + e) / (2 * b)

  if (max(abs(beta1 - beta1_old), abs(beta2 - beta2_old)) < tol) {
    cat("Converged after", i, "iterations\n")
    break
  }
}
```

```
## Converged after 5 iterations
```

```r
cat("Coordinate Descent Results:\n")
```

```
## Coordinate Descent Results:
```

```r
cat("beta1 =", beta1, "\n")
```

```
## beta1 = 0.6658955
```

```r
cat("beta2 =", beta2, "\n")
```

```
## beta2 = 0.4032028
```

```r
lm_model <- lm(y ~ x - 1)
cat("\nLM Function Results:\n")
```

```
##
## LM Function Results:
```

```r
print(coef(lm_model))
```

```
##        x1        x2
## 0.6658955 0.4032028
```