Stat 432 Homework 4

Assigned: Sep 16, 2024; Due: 11:59 PM CT, Sep 26, 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**, you are encouraged to document your comment on your experience with AI tools 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: Sparsity and Correlation

During our lecture, we considered a simulation model to analyze the variable selection property of Lasso. Now let's further investigate the prediction error of both Lasso and Ridge, and understand the bias-variance trade-off. Consider the linear model defined as:

$$Y = X^{\mathrm{T}}\boldsymbol{\beta} + \epsilon$$

Where $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{100})^T$ with $\beta_1 = \beta_{11} = \beta_{21} = \beta_{31} = 0.4$ and all other β parameters set to zero. The p-dimensional covariate X follows a multivariate Gaussian distribution:

$$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \Sigma_{p\times p}).$$

In Σ, all diagonal elements are 1, and all off-diagonal elements are ρ.

a. [15 points] A single Simulation Run

- Generate 200 training and 500 testing samples independently based on the above model.
- Use $\rho = 0.1$.
- Fit Lasso using `cv.glmnet()` on the training data with 10-fold cross-validation. Use `lambda.1se` to select the optimal λ.
- Report:
  - Prediction error (MSE) on the test data.
  - Report whether the true model was selected (you may refer to HW3 for this property).

```r
# Load necessary libraries
library(MASS)    # For multivariate normal distribution
library(glmnet)  # For Lasso regression
```

```
## Loading required package: Matrix

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

## Loaded glmnet 4.1-8
```

```r
# Set seed for reproducibility
set.seed(123)

# Simulation parameters
p <- 100                # Number of predictors
n_train <- 200          # Number of training samples
n_test <- 500           # Number of testing samples
rho <- 0.1              # Correlation coefficient

# True coefficient vector beta
beta <- rep(0, p)
beta[c(1, 11, 21, 31)] <- 0.4  # Non-zero coefficients

# Construct covariance matrix Sigma
Sigma <- matrix(rho, nrow = p, ncol = p)  # Initialize with rho
diag(Sigma) <- 1                          # Set diagonal elements to 1

# Generate training data
X_train <- mvrnorm(n = n_train, mu = rep(0, p), Sigma = Sigma)
epsilon_train <- rnorm(n_train, mean = 0, sd = 1)
Y_train <- X_train %*% beta + epsilon_train

# Generate testing data
X_test <- mvrnorm(n = n_test, mu = rep(0, p), Sigma = Sigma)
epsilon_test <- rnorm(n_test, mean = 0, sd = 1)
```

```r
Y_test <- X_test %*% beta + epsilon_test

# Fit Lasso model using 10-fold cross-validation
cv_lasso <- cv.glmnet(
  x = X_train,
  y = Y_train,
  alpha = 1,           # Lasso penalty
  nfolds = 10,         # Number of folds
  standardize = TRUE  # Standardize predictors
)

# Optimal lambda using lambda.1se
lambda_opt <- cv_lasso$lambda.1se

# Refit Lasso model with optimal lambda
lasso_model <- glmnet(
  x = X_train,
  y = Y_train,
  alpha = 1,
  lambda = lambda_opt,
  standardize = TRUE
)

# Predict on testing data
Y_pred <- predict(lasso_model, newx = X_test)

# Calculate Mean Squared Error (MSE)
MSE <- mean((Y_test - Y_pred)^2)
cat("Prediction error (MSE) on the test data:", round(MSE, 4), "\n")
```

```
## Prediction error (MSE) on the test data: 1.129
```

```r
# Extract coefficients from the model
coefficients <- as.vector(coef(lasso_model))
selected_variables <- which(coefficients != 0) - 1  # Subtract 1 for intercept

# Remove intercept from selected variables
selected_variables <- selected_variables[selected_variables != 0]
cat("Selected variables (indices):", selected_variables, "\n")
```

```
## Selected variables (indices): 1 11 12 21 25 30 31 33 43 93 99
```

```r
# Check if the true model was selected
true_variables <- c(1, 11, 21, 31)
if (identical(sort(selected_variables), true_variables)) {
  cat("True model was selected.\n")
} else {
  cat("True model was not fully selected.\n")
}
```

```
## True model was not fully selected.
```

b. [15 points] Higher Correlation and Multiple Simulation Runs

- Write a code to compare the previous simulation with $\rho = 0.1, 0.3, 0.5, 0.7, 0.9$.
- Perform 100 simulation runs as in part a) and record the prediction error and the status of the variable selection for Lasso.
- Report the average prediction error and the proportion of runs where the correct model was selected for each value of ρ.
- Discuss the reasons behind any observed changes.

```r
# Set seed for reproducibility
set.seed(123)

# Simulation parameters
p <- 100              # Number of predictors
n_train <- 200        # Number of training samples
n_test <- 500         # Number of testing samples
rho_values <- c(0.1, 0.3, 0.5, 0.7, 0.9)  # Correlation coefficients
n_simulations <- 100  # Number of simulation runs per rho

# True coefficient vector beta
beta <- rep(0, p)
beta[c(1, 11, 21, 31)] <- 0.4  # Non-zero coefficients
true_variables <- c(1, 11, 21, 31)

# Initialize lists to store results
results <- list()

# Loop over different rho values
for (rho in rho_values) {
  cat("Running simulations for rho =", rho, "\n")

  # Initialize vectors to store MSEs and model selection status
  MSEs <- numeric(n_simulations)
  model_selected <- logical(n_simulations)

  # Construct covariance matrix Sigma for current rho
  Sigma <- matrix(rho, nrow = p, ncol = p)  # Initialize with rho
  diag(Sigma) <- 1                          # Set diagonal elements to 1

  # Perform simulations
  for (sim in 1:n_simulations) {
    # Generate training data
    X_train <- mvrnorm(n = n_train, mu = rep(0, p), Sigma = Sigma)
    epsilon_train <- rnorm(n_train, mean = 0, sd = 1)
    Y_train <- X_train %*% beta + epsilon_train

    # Generate testing data
    X_test <- mvrnorm(n = n_test, mu = rep(0, p), Sigma = Sigma)
    epsilon_test <- rnorm(n_test, mean = 0, sd = 1)
    Y_test <- X_test %*% beta + epsilon_test

    # Fit Lasso model using 10-fold cross-validation
    cv_lasso <- cv.glmnet(
      x = X_train,
      y = Y_train,
      alpha = 1,          # Lasso penalty
      nfolds = 10,        # Number of folds
      standardize = TRUE,
      intercept = TRUE
    )

    # Optimal lambda using lambda.1se
    lambda_opt <- cv_lasso$lambda.1se

    # Refit Lasso model with optimal lambda
    lasso_model <- glmnet(
      x = X_train,
      y = Y_train,
      alpha = 1,
      lambda = lambda_opt,
      standardize = TRUE,
      intercept = TRUE
    )

    # Predict on testing data
    Y_pred <- predict(lasso_model, newx = X_test)

    # Calculate Mean Squared Error (MSE)
    MSEs[sim] <- mean((Y_test - Y_pred)^2)

    # Extract coefficients from the model
    coefficients <- as.vector(coef(lasso_model))
    selected_variables <- which(coefficients != 0) - 1  # Subtract 1 for intercept
    selected_variables <- selected_variables[selected_variables != 0]

    # Check if the true model was selected
    model_selected[sim] <- identical(sort(selected_variables), true_variables)
  }

  # Store results for current rho
  results[[as.character(rho)]] <- list(
    MSEs = MSEs,
    model_selected = model_selected,
    avg_MSE = mean(MSEs),
    prop_correct_model = mean(model_selected)
  )
}
```

```
## Running simulations for rho = 0.1
## Running simulations for rho = 0.3
## Running simulations for rho = 0.5
## Running simulations for rho = 0.7
## Running simulations for rho = 0.9
```

```
# Display results
cat("\nSummary of Results:\n")
```

```
##
```

```
## Summary of Results:
```

```
for (rho in rho_values) {
  res <- results[[as.character(rho)]]
  cat("rho =", rho, "\n")
  cat("Average Prediction Error (MSE):", round(res$avg_MSE, 4), "\n")
  cat("Proportion of Correct Model Selection:", round(res$prop_correct_model, 4), "\n\n")
}
```

```
## rho = 0.1
## Average Prediction Error (MSE): 1.1839
## Proportion of Correct Model Selection: 0.29
##
## rho = 0.3
## Average Prediction Error (MSE): 1.1822
## Proportion of Correct Model Selection: 0.06
##
## rho = 0.5
## Average Prediction Error (MSE): 1.193
## Proportion of Correct Model Selection: 0
##
## rho = 0.7
## Average Prediction Error (MSE): 1.177
## Proportion of Correct Model Selection: 0
##
## rho = 0.9
## Average Prediction Error (MSE): 1.1468
## Proportion of Correct Model Selection: 0
```

There are several reasons. First, Multicollinearity Effect: As p increases, the predictors become more highly correlated. High multicollinearity makes it difficult for Lasso to distinguish between the effects of correlated predictors.Lasso may select alternative predictors or distribute the coefficient values among correlated variables, leading to incorrect variable selection. Second, Bias-Variance Trade-off: With higher correlation, the variance of the estimators increases. Lasso tends to introduce bias to reduce variance, which may lead to excluding true predictors or including false ones. This trade-off affects both the prediction error and the variable selection performance.

c. [15 points] Ridge Regression

- Repeat task b) with the ridge regression. You do not need to record the variable selection status since ridge always select all variables.
- Report the average prediction error, do you see any difference between ridge and Lasso? Any performance differences within ridge regression as ρ changes?
- Discuss the reasons behind any observed changes.

```
# Set seed for reproducibility
set.seed(123)

# Simulation parameters
p <- 100              # Number of predictors
n_train <- 200        # Number of training samples
n_test <- 500         # Number of testing samples
rho_values <- c(0.1, 0.3, 0.5, 0.7, 0.9)  # Correlation coefficients
```

```r
n_simulations <- 100  # Number of simulation runs per rho

# True coefficient vector beta
beta <- rep(0, p)
beta[c(1, 11, 21, 31)] <- 0.4  # Non-zero coefficients

# Initialize lists to store results
ridge_results <- list()

# Loop over different rho values
for (rho in rho_values) {
  cat("Running simulations for rho =", rho, "\n")

  # Initialize vector to store MSEs
  MSEs <- numeric(n_simulations)

  # Construct covariance matrix Sigma for current rho
  Sigma <- matrix(rho, nrow = p, ncol = p)  # Initialize with rho
  diag(Sigma) <- 1                          # Set diagonal elements to 1

  # Perform simulations
  for (sim in 1:n_simulations) {
    # Generate training data
    X_train <- mvrnorm(n = n_train, mu = rep(0, p), Sigma = Sigma)
    epsilon_train <- rnorm(n_train, mean = 0, sd = 1)
    Y_train <- X_train %*% beta + epsilon_train

    # Generate testing data
    X_test <- mvrnorm(n = n_test, mu = rep(0, p), Sigma = Sigma)
    epsilon_test <- rnorm(n_test, mean = 0, sd = 1)
    Y_test <- X_test %*% beta + epsilon_test

    # Fit Ridge regression using cross-validation
    cv_ridge <- cv.glmnet(
      x = X_train,
      y = Y_train,
      alpha = 0,           # Ridge penalty
      nfolds = 10,         # Number of folds
      standardize = TRUE,
      intercept = TRUE
    )

    # Optimal lambda using lambda.1se
    lambda_opt <- cv_ridge$lambda.1se

    # Refit Ridge model with optimal lambda
    ridge_model <- glmnet(
      x = X_train,
      y = Y_train,
      alpha = 0,
      lambda = lambda_opt,
      standardize = TRUE,
      intercept = TRUE
    )

    # Predict on testing data
    Y_pred <- predict(ridge_model, newx = X_test)

    # Calculate Mean Squared Error (MSE)
    MSEs[sim] <- mean((Y_test - Y_pred)^2)
  }

  # Store results for current rho
  ridge_results[[as.character(rho)]] <- list(
    MSEs = MSEs,
    avg_MSE = mean(MSEs)
  )
}
```

```
## Running simulations for rho = 0.1
## Running simulations for rho = 0.3
## Running simulations for rho = 0.5
## Running simulations for rho = 0.7
## Running simulations for rho = 0.9
```

```r
# Display results
cat("\nSummary of Ridge Regression Results:\n")
```

```
##
## Summary of Ridge Regression Results:
```

```r
for (rho in rho_values) {
  res <- ridge_results[[as.character(rho)]]
  cat("rho =", rho, "\n")
  cat("Average Prediction Error (MSE):", round(res$avg_MSE, 4), "\n\n")
}
```

```
## rho = 0.1
## Average Prediction Error (MSE): 1.4446
##
## rho = 0.3
## Average Prediction Error (MSE): 1.4051
##
## rho = 0.5
## Average Prediction Error (MSE): 1.3499
##
## rho = 0.7
## Average Prediction Error (MSE): 1.2537
##
## rho = 0.9
## Average Prediction Error (MSE): 1.1499
```

At low p (low correlation), Lasso and Ridge have similar MSEs, with Ridge slightly better. At high p (high correlation), Ridge performs significantly better than Lasso in terms of prediction error. It is because that Ridge regression is known to handle multicollinearity effectively.By adding an L2 penalty, Ridge shrinks coefficients of correlated predictors together.his shrinkage reduces the variance of the estimates without adding significant bias.With higher correlation, the penalty effectively pools information from correlated predictors, improving prediction accuracy.

Question 2: Shrinkage Methods and Testing Error

In this question, we will predict the number of applications received using the variables in the College dataset that can be found in `ISLR2` package. The output variable will be the number of applications (Apps) and the other variables are predictors. If you use Python, consider migrating the data to an excel file and read it in Python.

a. [10 pts] Use the code below to divide the data set into a training set (600 observations) and a test set (177 observations). Fit a linear model (with all the input variables) using least squares on the training set using `lm()`, and report the test error (i.e., testing MSE).

```
  library(ISLR2)
```

```
##
## Attaching package: 'ISLR2'

## The following object is masked from 'package:MASS':
##
##     Boston
```

```
  data(College)

  # generate the indices for the testing data
  set.seed(7)
  test_idx = sample(nrow(College), 177)
  train = College[-test_idx,]
  test = College[test_idx,]
```

```
library(ISLR2)
data(College)

# Generate the indices for the testing data
set.seed(7)
test_idx = sample(nrow(College), 177)
train = College[-test_idx, ]
test = College[test_idx, ]

# Fit a linear model using least squares on the training set
lm_fit <- lm(Apps ~ ., data = train)

# Predict the number of applications on the test set
predictions <- predict(lm_fit, newdata = test)

# Compute the test Mean Squared Error (MSE)
test_MSE <- mean((predictions - test$Apps)^2)

# Report the test MSE
print(paste("Test MSE:", test_MSE))
```

```
## [1] "Test MSE: 961142.269019084"
```

b. [10 pts] Compare Lasso and Ridge regression on this problem. Train the model using cross-validation on the training set. Report the test error for both Lasso and Ridge regression. Use `lambda.min` and `lambda.1se` to select the optimal λ for both methods.

```
x_train = model.matrix(Apps ~ ., data = train)[, -1]
y_train = train$Apps

x_test = model.matrix(Apps ~ ., data = test)[, -1]
y_test = test$Apps

# Ridge Regression with Cross-Validation
set.seed(1)
cv_ridge = cv.glmnet(x_train, y_train, alpha = 0, nfolds = 10)

# Optimal lambda values for Ridge Regression
lambda_min_ridge = cv_ridge$lambda.min
lambda_1se_ridge = cv_ridge$lambda.1se

# Predictions using lambda.min
pred_ridge_min = predict(cv_ridge, s = lambda_min_ridge, newx = x_test)
mse_ridge_min = mean((pred_ridge_min - y_test)^2)

# Predictions using lambda.1se
pred_ridge_1se = predict(cv_ridge, s = lambda_1se_ridge, newx = x_test)
mse_ridge_1se = mean((pred_ridge_1se - y_test)^2)

# Lasso Regression with Cross-Validation
set.seed(1)
cv_lasso = cv.glmnet(x_train, y_train, alpha = 1, nfolds = 10)

# Optimal lambda values for Lasso Regression
lambda_min_lasso = cv_lasso$lambda.min
lambda_1se_lasso = cv_lasso$lambda.1se

# Predictions using lambda.min
pred_lasso_min = predict(cv_lasso, s = lambda_min_lasso, newx = x_test)
mse_lasso_min = mean((pred_lasso_min - y_test)^2)

# Predictions using lambda.1se
pred_lasso_1se = predict(cv_lasso, s = lambda_1se_lasso, newx = x_test)
mse_lasso_1se = mean((pred_lasso_1se - y_test)^2)

# Report the test errors
cat("Ridge Regression Test MSE (lambda.min):", round(mse_ridge_min, 2), "\n")
```

```
## Ridge Regression Test MSE (lambda.min): 875548.9
```

```
cat("Ridge Regression Test MSE (lambda.1se):", round(mse_ridge_1se, 2), "\n")
```

```
## Ridge Regression Test MSE (lambda.1se): 1338040
```

```r
cat("Lasso Regression Test MSE (lambda.min):", round(mse_lasso_min, 2), "\n")
```

```
## Lasso Regression Test MSE (lambda.min): 948102.7
```

```r
cat("Lasso Regression Test MSE (lambda.1se):", round(mse_lasso_1se, 2), "\n")
```

```
## Lasso Regression Test MSE (lambda.1se): 1076206
```

c. [20 pts] The `glmnet` package implemented a new feature called `relaxed` fits and the associated tuning parameter `gamma`. You can find some brief explaination of this feature at the documentation of this package. See

   - CRAN Documentation
   - glmnet Vignette

   Read these documentations regarding the `gamma` parameter, and summarize the idea of this feature in terms of the loss function being used. You need to write it specifically in terms of the data vectors $\mathbf{y}$ and matrix $\mathbf{X}$ and define any notations you need. Only consider the Lasso penalty for this question.

   After this, implement this feature and utilize the cross-validation to find the optimal λ and γ for the College dataset. Report the test error for the optimal model.

```r
set.seed(1)

# Define a sequence of gamma values
gamma_values = seq(0, 1, by = 0.1)

# Cross-validation with relaxed lasso
cv_relaxed = cv.glmnet(
  x_train, y_train,
  alpha = 1,
  nfolds = 10,
  relax = TRUE,
  gamma = gamma_values,
  type.measure = "mse"
)

# Extract the optimal lambda and gamma values
# Use the stored 'lambda.min' and 'gamma.min' from the cv.glmnet object
optimal_lambda = cv_relaxed$lambda.min
optimal_gamma = cv_relaxed$relaxed$gamma.min

# Predict on test set using the optimal lambda and gamma
pred_relaxed = predict(
  cv_relaxed,
  newx = x_test,
  s = "lambda.min",
  gamma = "gamma.min"
)

# Compute the test Mean Squared Error (MSE)
mse_relaxed = mean((pred_relaxed - y_test)^2)
```

```
# Report the test error and optimal parameters
cat("Relaxed Lasso Test MSE:", round(mse_relaxed, 2), "\n")
```

```
## Relaxed Lasso Test MSE: 1019309
```

```
cat("Optimal lambda:", optimal_lambda, "\n")
```

```
## Optimal lambda: 1.772301
```

```
cat("Optimal gamma:", optimal_gamma, "\n")
```

```
## Optimal gamma: 0
```

Question 3: Penalized Logistic Regression

In HW3, we used `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). The outcome `golub.cl` is an indicator for two leukemia types: Acute Lymphoblastic Leukemia (ALL) or Acute Myeloid Leukemia (AML). In genetic analysis, many gene expressions are highly correlated. Hence we could consider the Elastic net model for both sparsity and correlation.

```
  if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
  BiocManager::install("multtest")
```

[15 pts] Fit logistic regression to this dataset. Use a grid of α values in $[0, 1]$ and report the best α and λ values using cross-validation.

```
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)

# Prepare the data
X <- t(golub)  # Transpose to have samples as rows and genes as columns
y <- golub.cl  # Class labels: 0 for ALL, 1 for AML

# Ensure the response variable is a factor
y <- as.factor(y)

# Define a grid of alpha values from 0 to 1
alpha_grid <- seq(0, 1, length = 11)  # Generates 11 values: 0, 0.1, ..., 1

# Initialize vectors to store cross-validation errors and corresponding lambda values
cv_errors <- rep(NA, length(alpha_grid))
lambda_min <- rep(NA, length(alpha_grid))

# Set seed for reproducibility
set.seed(123)

# Loop over each alpha value and perform cross-validation
for (i in 1:length(alpha_grid)) {
  alpha_val <- alpha_grid[i]
  cv_model <- cv.glmnet(X, y, alpha = alpha_val, family = "binomial", type.measure = "class")
  cv_errors[i] <- min(cv_model$cvm)
  lambda_min[i] <- cv_model$lambda.min
}

# Identify the alpha with the minimum cross-validation error
best_index <- which.min(cv_errors)
best_alpha <- alpha_grid[best_index]
best_lambda <- lambda_min[best_index]

# Report the best alpha and lambda values
cat("Best alpha:", best_alpha, "\n")
```

```
## Best alpha: 0
```

```
cat("Best lambda:", best_lambda, "\n")
```

```
## Best lambda: 10.89235
```