Stat 432 Homework 4

Assigned: Sep 16, 2024; Due: 11:59 PM CT, Sep 26, 2024

Contents

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
library(MASS)        # For mvrnorm function
library(glmnet)      # For Lasso regression
```

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

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

```r
library(leaps)       # For subset selection in HW3 comparison
```

```
## Warning: package 'leaps' was built under R version 4.3.3
```

```r
set.seed(432)

# Define parameters
n_train <- 200
n_test <- 500
p <- 100
rho <- 0.1
true_beta <- rep(0, p)
true_beta[1] <- 0.4  # Set the non-zero beta values
true_beta[11] <- 0.4
true_beta[21] <- 0.4
true_beta[31] <- 0.4

# Covariance matrix with rho on the off-diagonals and 1 on the diagonal
Sigma <- matrix(rho, nrow = p, ncol = p)
diag(Sigma) <- 1

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

# Generate test data
X_test <- mvrnorm(n_test, mu = rep(0, p), Sigma = Sigma)
epsilon_test <- rnorm(n_test)
Y_test <- X_test %*% true_beta + epsilon_test

# Fit Lasso using cross-validation
cv_lasso <- cv.glmnet(X_train, Y_train, alpha = 1, nfolds = 10)

# Extract the optimal lambda using the "lambda.1se" rule
lambda_opt <- cv_lasso$lambda.1se

# Fit the Lasso model using the optimal lambda
lasso_model <- glmnet(X_train, Y_train, alpha = 1, lambda = lambda_opt)

# Prediction on test data
Y_pred <- predict(lasso_model, newx = X_test)

# Calculate Mean Squared Error on the test data
mse_test <- mean((Y_test - Y_pred)^2)
print(paste("Test MSE:", mse_test))
```

```
## [1] "Test MSE: 1.16548440945485"
```

```r
# Check if the true model was selected (use nonzero coefficients in lasso)
selected_variables <- which(coef(lasso_model) != 0)
selected_variables <- selected_variables[-1] - 1
```

```r
true_model_selected <- all(selected_variables %in% c(1, 11, 21, 31)) && length(selected_variables) == 4

true_model_selected
```

```
## [1] FALSE
```

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(432)

# Define parameters
n_train <- 200
n_test <- 500
p <- 100
true_beta <- rep(0, p)
true_beta[1] <- 0.4  # Set the non-zero beta values
true_beta[11] <- 0.4
true_beta[21] <- 0.4
true_beta[31] <- 0.4
num_simulations <- 100
rhos <- c(0.1, 0.3, 0.5, 0.7, 0.9)

# Initialize storage for results
results <- data.frame(rho = rhos, avg_mse = rep(0, length(rhos)), correct_model_proportion = rep(0, leng

# Function to generate covariance matrix
generate_cov_matrix <- function(rho, p) {
  Sigma <- matrix(rho, nrow = p, ncol = p)
  diag(Sigma) <- 1
  return(Sigma)
}

# Main simulation loop
for (rho_index in 1:length(rhos)) {

  rho <- rhos[rho_index]

  # Initialize counters
  total_mse <- 0
  correct_model_count <- 0

  for (sim in 1:num_simulations) {

    # Generate covariance matrix
    Sigma <- generate_cov_matrix(rho, p)
    # Generate training data
    X_train <- mvrnorm(n_train, mu = rep(0, p), Sigma = Sigma)
    epsilon_train <- rnorm(n_train)
    Y_train <- X_train %*% true_beta + epsilon_train

    # Generate test data
    X_test <- mvrnorm(n_test, mu = rep(0, p), Sigma = Sigma)
    epsilon_test <- rnorm(n_test)
    Y_test <- X_test %*% true_beta + epsilon_test

    # Fit Lasso using cross-validation
    cv_lasso <- cv.glmnet(X_train, Y_train, alpha = 1, nfolds = 10)

    # Extract the optimal lambda using the "lambda.1se" rule
    lambda_opt <- cv_lasso$lambda.1se

    # Fit the Lasso model using the optimal lambda
    lasso_model <- glmnet(X_train, Y_train, alpha = 1, lambda = lambda_opt)

    # Prediction on test data
    Y_pred <- predict(lasso_model, newx = X_test)

    # Calculate Mean Squared Error on the test data
    mse_test <- mean((Y_test - Y_pred)^2)
    total_mse <- total_mse + mse_test

    # Check if the true model was selected
    selected_variables <- which(coef(lasso_model) != 0)
    selected_variables <- selected_variables[-1] - 1
    true_model_selected <- all(selected_variables %in% c(1, 11, 21, 31)) && length(selected_variables)

    if (true_model_selected) {
      correct_model_count <- correct_model_count + 1
    }

  }

  # Store results for the current value of rho
  avg_mse <- total_mse / num_simulations
  correct_model_proportion <- correct_model_count / num_simulations

  results$avg_mse[rho_index] <- avg_mse
  results$correct_model_proportion[rho_index] <- correct_model_proportion
}

# Print the results
print(results)
```

```
##   rho  avg_mse correct_model_proportion
## 1 0.1 1.191106                     0.30
## 2 0.3 1.212688                     0.09
## 3 0.5 1.169009                     0.02
## 4 0.7 1.157866                     0.00
## 5 0.9 1.157238                    0.00
```

The correct model proportion decrease as rho increases. As the correlation between predictors increases, it becomes harder for LASSO to distinguish between them. LASSO tends to pick one of the highly correlated predictors while shrinking the coefficients of others to zero, which may result in selecting an incorrect model. In this case, as rho increases, LASSO struggles to select the true underlying model, and the overall fit (measured by MSE) improves a little only.

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.

```r
set.seed(432)

# Define parameters
n_train <- 200
n_test <- 500
p <- 100
true_beta <- rep(0, p)
true_beta[1] <- 0.4  # Set the non-zero beta values
true_beta[11] <- 0.4
true_beta[21] <- 0.4
true_beta[31] <- 0.4
num_simulations <- 100
rhos <- c(0.1, 0.3, 0.5, 0.7, 0.9)

# Initialize storage for results
results_ridge <- data.frame(rho = rhos, avg_mse = rep(0, length(rhos)))

# Function to generate covariance matrix
generate_cov_matrix <- function(rho, p) {
  Sigma <- matrix(rho, nrow = p, ncol = p)
  diag(Sigma) <- 1
  return(Sigma)
}

# Main simulation loop for Ridge regression
for (rho_index in 1:length(rhos)) {

  rho <- rhos[rho_index]

  # Initialize counters
  total_mse <- 0

  for (sim in 1:num_simulations) {

    # Generate covariance matrix
    Sigma <- generate_cov_matrix(rho, p)
    # Generate training data
    X_train <- mvrnorm(n_train, mu = rep(0, p), Sigma = Sigma)
    epsilon_train <- rnorm(n_train)
    Y_train <- X_train %*% true_beta + epsilon_train

    # Generate test data
    X_test <- mvrnorm(n_test, mu = rep(0, p), Sigma = Sigma)
    epsilon_test <- rnorm(n_test)
    Y_test <- X_test %*% true_beta + epsilon_test

    # Fit Ridge using cross-validation (alpha = 0 for Ridge)
    cv_ridge <- cv.glmnet(X_train, Y_train, alpha = 0, nfolds = 10)

    # Extract the optimal lambda using the "lambda.1se" rule
    lambda_opt <- cv_ridge$lambda.1se

    # Fit the Ridge model using the optimal lambda
    ridge_model <- glmnet(X_train, Y_train, alpha = 0, lambda = lambda_opt)

    # Prediction on test data
    Y_pred <- predict(ridge_model, newx = X_test)

    # Calculate Mean Squared Error on the test data
    mse_test <- mean((Y_test - Y_pred)^2)
    total_mse <- total_mse + mse_test
  }

  # Store average MSE for the current value of rho
  avg_mse <- total_mse / num_simulations
  results_ridge$avg_mse[rho_index] <- avg_mse
}

# Print the results for Ridge regression
print(results_ridge)
```

```
##   rho  avg_mse
## 1 0.1 1.459797
## 2 0.3 1.425839
## 3 0.5 1.327182
## 4 0.7 1.237139
## 5 0.9 1.155435
```

Compared to LASSO, Ridge regression MSE showed a sharper decline as rho increased. This is expected since Ridge applies a smooth regularization (L2 penalty) that shrinks all coefficients more evenly, in contrast to LASSO's tendency to shrink some coefficients to zero. Ridge regression handles multicollinearity well by shrinking the coefficients of correlated variables without setting them exactly to zero. As the correlation between predictors increases, Ridge regression can distribute the effect of correlated predictors, improving the model's fit and reducing the MSE. This explains why the avg_mse decreases as the correlation among predictors increases.

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).

```r
library(ISLR2)
```

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

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

```r
data(College)

# Set the seed for reproducibility
set.seed(7)

# Split the data into training and testing sets
test_idx <- sample(nrow(College), 177)  # Select 177 random rows for testing
train <- College[-test_idx, ]           # Training data (excluding test indices)
test <- College[test_idx, ]             # Test data (selected indices)

# Fit the linear model using all predictors in the training data
linear_model <- lm(Apps ~ ., data = train)

# Make predictions on the test set
predictions <- predict(linear_model, newdata = test)

# Calculate the Mean Squared Error (MSE) on the test set
mse_test <- mean((test$Apps - predictions)^2)

# Print the test error (MSE)
print(paste("Test MSE:", mse_test))
```

```
## [1] "Test MSE: 961142.269019081"
```

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.

```r
library(ISLR2)
library(glmnet)
```

```r
# Load the College dataset
data(College)

# Set the seed for reproducibility
set.seed(7)

# Split the data into training and testing sets
test_idx <- sample(nrow(College), 177)
train <- College[-test_idx, ]
test <- College[test_idx, ]

# Prepare the data for glmnet (needs matrix form)
X_train <- model.matrix(Apps ~ ., data = train)[, -1]  # Exclude the intercept
Y_train <- train$Apps

X_test <- model.matrix(Apps ~ ., data = test)[, -1]    # Exclude the intercept
Y_test <- test$Apps



# Fit Lasso model using cross-validation (alpha = 1 for Lasso)
cv_lasso <- cv.glmnet(X_train, Y_train, alpha = 1, nfolds = 10)

# Predictions using lambda.min and lambda.1se for Lasso
lasso_pred_min <- predict(cv_lasso, newx = X_test, s = "lambda.min")
lasso_pred_1se <- predict(cv_lasso, newx = X_test, s = "lambda.1se")

# Calculate the test MSE for Lasso
mse_lasso_min <- mean((Y_test - lasso_pred_min)^2)
mse_lasso_1se <- mean((Y_test - lasso_pred_1se)^2)



# Fit Ridge model using cross-validation (alpha = 0 for Ridge)
cv_ridge <- cv.glmnet(X_train, Y_train, alpha = 0, nfolds = 10)

# Predictions using lambda.min and lambda.1se for Ridge
ridge_pred_min <- predict(cv_ridge, newx = X_test, s = "lambda.min")
ridge_pred_1se <- predict(cv_ridge, newx = X_test, s = "lambda.1se")

# Calculate the test MSE for Ridge
mse_ridge_min <- mean((Y_test - ridge_pred_min)^2)
mse_ridge_1se <- mean((Y_test - ridge_pred_1se)^2)



# Print the test MSE for both Lasso and Ridge
cat("Lasso Regression:\n")
```

```
## Lasso Regression:
```

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

```
## Test MSE (lambda.min): 906714.6
```

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

```
## Test MSE (lambda.1se): 1116979
```

```r
cat("Ridge Regression:\n")
```

```
## Ridge Regression:
```

```r
cat("Test MSE (lambda.min):", mse_ridge_min, "\n")
```

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

```r
cat("Test MSE (lambda.1se):", mse_ridge_1se, "\n")
```

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

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
data(College)

# Set the seed for reproducibility
set.seed(7)

# Split the data into training and testing sets
test_idx <- sample(nrow(College), 177)  # Select 177 random rows for testing
train <- College[-test_idx, ]           # Training data (excluding test indices)
test <- College[test_idx, ]             # Test data (selected indices)

# Prepare the data for glmnet (needs matrix form)
X_train <- model.matrix(Apps ~ ., data = train)[, -1]  # Exclude the intercept
Y_train <- train$Apps

X_test <- model.matrix(Apps ~ ., data = test)[, -1]    # Exclude the intercept
Y_test <- test$Apps

# Fit Relaxed Lasso model using cross-validation
cv_relaxed_lasso <- cv.glmnet(X_train, Y_train, alpha = 1, relax = TRUE)
```

```r
# Get the best lambda and gamma values using "lambda.min" and "gamma.min"
lambda_opt <- cv_relaxed_lasso$lambda.min
gamma_opt <- cv_relaxed_lasso$relaxed$gamma.min

# Make predictions on the test set using the optimal lambda and gamma
predictions <- predict(cv_relaxed_lasso, newx = X_test, s = "lambda.min", gamma = "gamma.min")

# Calculate the Mean Squared Error (MSE) on the test set
mse_test <- mean((Y_test - predictions)^2)

# Print the test error
print(paste("Test MSE for Relaxed Lasso (optimal lambda and gamma):", mse_test))
```

```
## [1] "Test MSE for Relaxed Lasso (optimal lambda and gamma): 1019308.41243271"
```

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.

```r
  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.

```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")'.
```

```r
library(glmnet)

data(golub)
X <- t(golub)
Y <- as.factor(golub.cl)

Y <- as.numeric(Y) - 1  # Convert factor to 0/1

# Set up a grid of alpha values for Elastic Net
alpha_grid <- seq(0, 1, by = 0.1)  # Alpha values from 0 to 1

# Initialize placeholders for storing the best results
best_alpha <- NA
best_lambda <- NA
best_mse <- Inf
best_cvfit <- NULL

# Iterate over the grid of alpha values
for (alpha_value in alpha_grid) {

  # Perform cross-validation for logistic regression with Elastic Net
  cvfit <- cv.glmnet(X, Y, family = "binomial", alpha = alpha_value)

  # Get the minimum lambda for the current alpha
  lambda_min <- cvfit$lambda.min

  # Get the cross-validated mean squared error for the current model
  mse <- cvfit$cvm[cvfit$lambda == lambda_min]

  # Check if this is the best model so far
  if (mse < best_mse) {
    best_mse <- mse
    best_alpha <- alpha_value
    best_lambda <- lambda_min
    best_cvfit <- cvfit
  }
}
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```r
# Print the best alpha and lambda
cat("Best Alpha:", best_alpha, "\n")
```

```
## Best Alpha: 1
```

```r
cat("Best Lambda:", best_lambda, "\n")
```

```
## Best Lambda: 0.003914509
```