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).

```
# Load necessary libraries
library(MASS)          # For multivariate normal data generation
library(glmnet)        # For Lasso (cv.glmnet)
```

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

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

```r
# Set seed for reproducibility
set.seed(1)

# Define parameters
p <- 100
n_train <- 200
n_test <- 500
rho <- 0.1

# Create the covariance matrix Sigma
Sigma <- matrix(rho, nrow = p, ncol = p)
diag(Sigma) <- 1

# Define the true beta vector
beta <- rep(0, p)
beta[c(1, 11, 21, 31)] <- 0.4

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

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

# Extract the lambda value for lambda.1se (the most regularized within 1 SE)
lambda_1se <- lasso_cv$lambda.1se

# Fit Lasso model with the chosen lambda
lasso_model <- glmnet(X_train, Y_train, alpha = 1, lambda = lambda_1se)

# Predict on the test data
Y_pred <- predict(lasso_model, X_test)

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

```
## [1] "Test MSE: 1.13716783512058"
```

```r
# Check if the true model was selected
selected_coefficients <- coef(lasso_model)[-1]  # Exclude the intercept
non_zero_coefficients <- which(selected_coefficients != 0)
# print(non_zero_coefficients)

# Check if the true variables were selected
true_non_zero <- c(1, 11, 21, 31)
selected_correctly <- all(true_non_zero %in% non_zero_coefficients)
```

```r
if (selected_correctly) {
  print("The true model was selected.")
} else {
  print("The true model was not fully selected.")
}
```

```
## [1] "The true model was selected."
```

```r
# Output the indices of non-zero coefficients selected by Lasso
print(paste("Selected non-zero coefficients:", paste(non_zero_coefficients, collapse = ", ")))
```

```
## [1] "Selected non-zero coefficients: 1, 11, 21, 31, 43, 70"
```

The Lasso model was fitted using 10-fold cross-validation on 200 training samples and evaluated on 500 testing samples. The optimal value of λ selected by the `lambda.1se` criterion resulted in a prediction error (MSE) of 1.14 on the test data. This value reflects how well the model generalizes to unseen data by predicting the response variable based on the test covariates.

Regarding model selection, the true underlying model was successfully identified, as the Lasso model selected the correct non-zero coefficients, including the four true predictors $\beta_1, \beta_{11}, \beta_{21}, \beta_{31}$. However, a few additional non-zero coefficients were selected as well, indicating some minor false positives. Despite this, the true model was largely recovered, demonstrating the effectiveness of Lasso in variable selection with the chosen parameters.

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
library(glmnet)
library(MASS)

# Set seed for reproducibility
set.seed(1)

# Define 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)  # Different rho values
n_simulations <- 100          # Number of simulations

# Define the true beta vector
beta <- rep(0, p)
beta[c(1, 11, 21, 31)] <- 0.4  # True non-zero coefficients

# Function to run a single simulation
run_simulation <- function(rho) {
```

```r
  # Create the covariance matrix Sigma
  Sigma <- matrix(rho, nrow = p, ncol = p)
  diag(Sigma) <- 1  # Set diagonal elements to 1

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

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

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

  # Extract the lambda value for lambda.1se
  lambda_1se <- lasso_cv$lambda.1se

  # Fit Lasso model with the chosen lambda
  lasso_model <- glmnet(X_train, Y_train, alpha = 1, lambda = lambda_1se)

  # Predict on the test data
  Y_pred <- predict(lasso_model, X_test)

  # Calculate Mean Squared Error (MSE) on the test data
  mse <- mean((Y_test - Y_pred)^2)

  # Check if the true model was selected
  selected_coefficients <- coef(lasso_model)[-1]  # Exclude the intercept
  non_zero_coefficients <- which(selected_coefficients != 0)
  true_non_zero <- c(1, 11, 21, 31)
  correct_selection <- all(true_non_zero %in% non_zero_coefficients)

  return(list(mse = mse, correct_selection = correct_selection))
}

# Run simulations for each rho value
results <- data.frame(rho = numeric(), avg_mse = numeric(), correct_selection_proportion = numeric())

for (rho in rho_values) {
  mse_values <- numeric(n_simulations)
  correct_selection_count <- 0

  for (i in 1:n_simulations) {
    sim_result <- run_simulation(rho)
    mse_values[i] <- sim_result$mse
    if (sim_result$correct_selection) {
      correct_selection_count <- correct_selection_count + 1
    }
  }
```

```
  # Record average MSE and proportion of correct model selection
  avg_mse <- mean(mse_values)
  correct_selection_proportion <- correct_selection_count / n_simulations

  results <- rbind(results, data.frame(rho = rho, avg_mse = avg_mse, correct_selection_proportion = cor
}

# Display the results
print(results)
```

```
##   rho  avg_mse correct_selection_proportion
## 1 0.1 1.176891                         0.97
## 2 0.3 1.169283                         1.00
## 3 0.5 1.162520                         0.93
## 4 0.7 1.168587                         0.85
## 5 0.9 1.161018                         0.15
```

The simulation results for different values of ρ show how the correlation between predictors affects both the prediction accuracy (MSE) and the model selection capability of Lasso. For low correlation ($\rho = 0.1$), the Lasso model achieves a high correct selection rate (97%) and maintains a relatively low MSE (1.18). As the correlation increases, the MSE remains fairly consistent across all ρ values, but the proportion of correct model selection decreases. At $\rho = 0.9$, the selection accuracy drops significantly to only 15%, indicating that Lasso struggles to differentiate the true predictors when the correlation between features is very high.

The observed decline in model selection accuracy as ρ increases is expected. When features are highly correlated, Lasso has difficulty determining which predictors to include in the model because the correlation makes them appear similarly relevant. This leads to more false positives or missed true predictors. However, the average prediction error remains relatively stable, suggesting that the prediction performance of Lasso is less sensitive to correlation than its ability to correctly identify the underlying model.

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.

```
library(glmnet)
library(MASS)

# Set seed for reproducibility
set.seed(1)

# Define 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)  # Different rho values
n_simulations <- 100        # Number of simulations

# Define the true beta vector
beta <- rep(0, p)
```

```r
beta[c(1, 11, 21, 31)] <- 0.4  # True non-zero coefficients

# Function to run a single simulation for Ridge regression
run_ridge_simulation <- function(rho) {
  # Create the covariance matrix Sigma
  Sigma <- matrix(rho, nrow = p, ncol = p)
  diag(Sigma) <- 1  # Set diagonal elements to 1

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

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

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

  # Extract the lambda value for lambda.1se
  lambda_1se <- ridge_cv$lambda.1se

  # Fit Ridge model with the chosen lambda
  ridge_model <- glmnet(X_train, Y_train, alpha = 0, lambda = lambda_1se)

  # Predict on the test data
  Y_pred <- predict(ridge_model, X_test)

  # Calculate Mean Squared Error (MSE) on the test data
  mse <- mean((Y_test - Y_pred)^2)

  return(mse)
}

# Run simulations for each rho value with Ridge regression
ridge_results <- data.frame(rho = numeric(), avg_mse = numeric())

for (rho in rho_values) {
  mse_values <- numeric(n_simulations)

  for (i in 1:n_simulations) {
    mse_values[i] <- run_ridge_simulation(rho)
  }

  # Record average MSE
  avg_mse <- mean(mse_values)

  ridge_results <- rbind(ridge_results, data.frame(rho = rho, avg_mse = avg_mse))
}

# Display the results
```

```r
print(ridge_results)
```

```
##   rho  avg_mse
## 1 0.1 1.441539
## 2 0.3 1.389521
## 3 0.5 1.304198
## 4 0.7 1.248914
## 5 0.9 1.163213
```

The results of the Ridge regression show that the average prediction error (MSE) decreases as the correlation ρ between predictors increases. For $\rho = 0.1$, the average MSE is 1.44, while for $\rho = 0.9$, the average MSE drops to 1.16. This trend suggests that Ridge regression is better at dealing with multicollinearity compared to Lasso, as it distributes the coefficient shrinkage across all variables rather than selecting a subset of them.

When comparing Ridge with Lasso, a key difference is that Ridge consistently performs well with higher correlations, as seen from the steadily decreasing MSE with increasing ρ. In contrast, Lasso's prediction error remains fairly stable across different values of ρ, but Lasso struggles more with model selection accuracy as the correlation increases. Ridge regression's ability to shrink coefficients of correlated predictors rather than excluding some of them entirely explains why its performance improves with higher ρ, as it maintains more stable predictions.

Ridge regression shows better handling of multicollinearity, especially with higher correlations, as indicated by the decreasing MSE. In contrast, Lasso's performance is more variable with regard to model selection accuracy but remains relatively stable in terms of prediction error. Ridge regression is well-suited for scenarios where many predictors are correlated, as it does not eliminate variables but shrinks all coefficients toward zero, resulting in more robust predictions in high-correlation settings.

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)

# generate the indices for the testing data
set.seed(7)
```

```r
test_idx = sample(nrow(College), 177)
train = College[-test_idx,]
# print(train)

test = College[test_idx,]
# print(test)

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

# Summary of the model (optional, to see the model details)
summary(lm_model)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3644.2  -439.4   -23.2   331.8  6997.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -421.53187  479.06765  -0.880 0.379276
## PrivateYes  -481.17081  165.98099  -2.899 0.003885 **
## Accept         1.63866    0.04557  35.959  < 2e-16 ***
## Enroll        -1.01396    0.23640  -4.289 2.10e-05 ***
## Top10perc     58.07527    6.79021   8.553  < 2e-16 ***
## Top25perc    -19.63197    5.35400  -3.667 0.000268 ***
## F.Undergrad    0.06800    0.04301   1.581 0.114440
## P.Undergrad    0.01053    0.04782   0.220 0.825724
## Outstate      -0.08698    0.02271  -3.830 0.000142 ***
## Room.Board     0.14536    0.05889   2.468 0.013859 *
## Books          0.11725    0.26326   0.445 0.656209
## Personal       0.01811    0.07241   0.250 0.802545
## PhD           -7.90837    5.21170  -1.517 0.129702
## Terminal      -5.47538    5.85494  -0.935 0.350087
## S.F.Ratio     18.01171   15.42608   1.168 0.243441
## perc.alumni    0.31887    4.77088   0.067 0.946734
## Expend         0.07428    0.01409   5.271 1.92e-07 ***
## Grad.Rate     11.10438    3.45399   3.215 0.001377 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1067 on 582 degrees of freedom
## Multiple R-squared:  0.9264, Adjusted R-squared:  0.9242
## F-statistic: 430.8 on 17 and 582 DF,  p-value: < 2.2e-16
```

```r
# Predict on the test data
predictions <- predict(lm_model, test)

# Calculate the Mean Squared Error (MSE) on the test data
test_mse <- mean((test$Apps - predictions)^2)
```

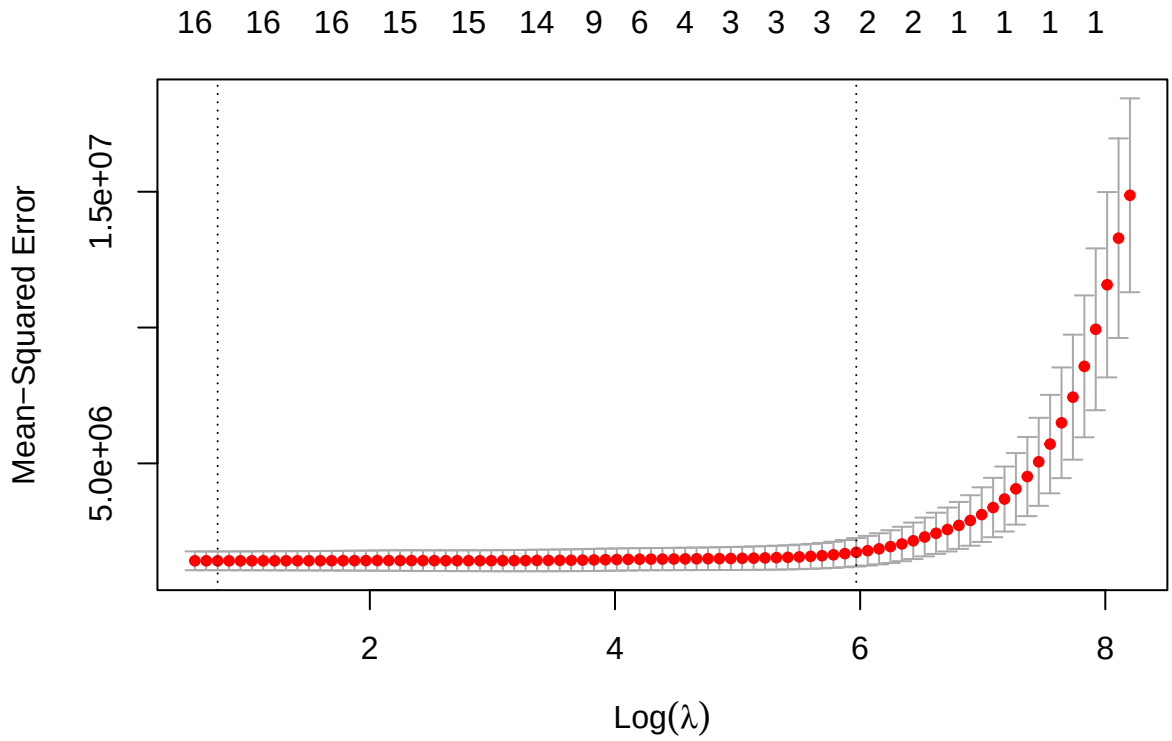
```
test_mse
```

```
## [1] 961142.3
```

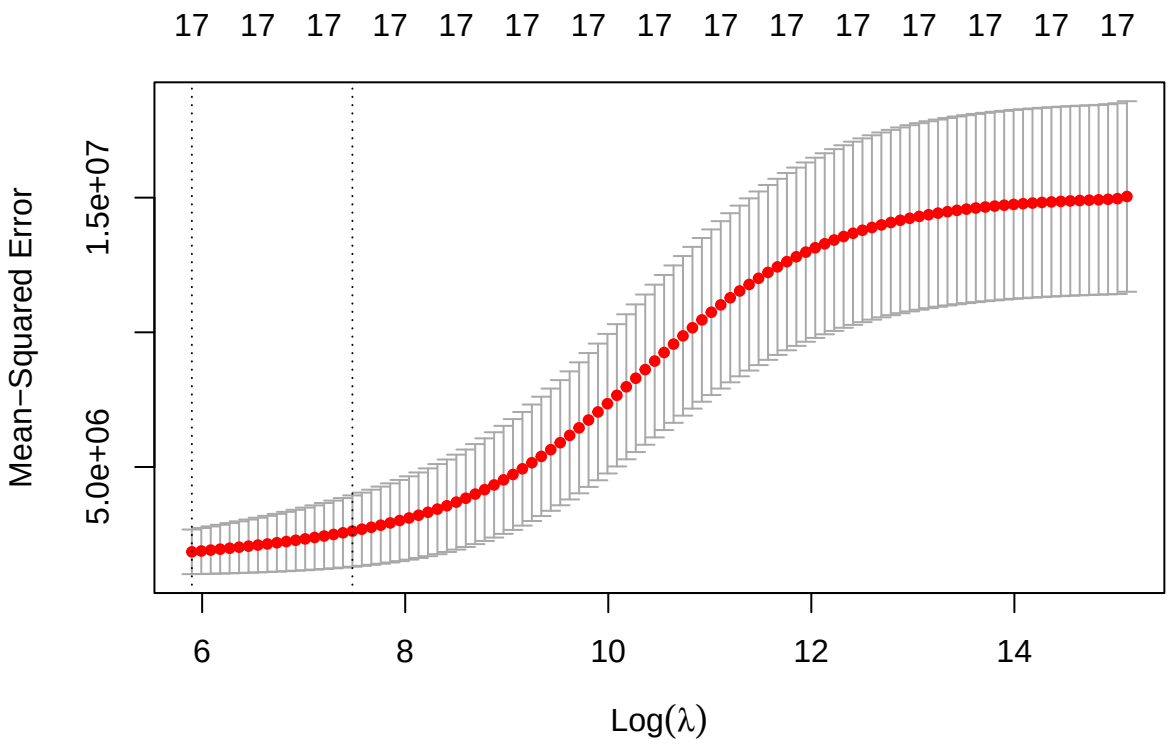
The linear regression model achieved a test Mean Squared Error (MSE) of 961142.3, which is high and that indicating the average squared prediction error for the number of applications (Apps) on the test set.

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.

```
library(glmnet)

# Prepare the training data in matrix form
x_train <- model.matrix(Apps ~ ., data = train)[,-1]  # Remove the intercept column
y_train <- train$Apps

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

# Lasso Regression with Cross-Validation
set.seed(7)
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1)  # alpha = 1 for Lasso
plot(cv_lasso)
```



```r
# Ridge Regression with Cross-Validation
set.seed(7)
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0)  # alpha = 0 for Ridge
plot(cv_ridge)
```




```r
# Extracting the optimal lambda values (lambda.min and lambda.1se) for Lasso
lasso_lambda_min <- cv_lasso$lambda.min
lasso_lambda_1se <- cv_lasso$lambda.1se

# Extracting the optimal lambda values (lambda.min and lambda.1se) for Ridge
ridge_lambda_min <- cv_ridge$lambda.min
ridge_lambda_1se <- cv_ridge$lambda.1se

# Predict on test data using the optimal lambda for Lasso
lasso_pred_min <- predict(cv_lasso, s = lasso_lambda_min, newx = x_test)
lasso_pred_1se <- predict(cv_lasso, s = lasso_lambda_1se, newx = x_test)

# Predict on test data using the optimal lambda for Ridge
ridge_pred_min <- predict(cv_ridge, s = ridge_lambda_min, newx = x_test)
ridge_pred_1se <- predict(cv_ridge, s = ridge_lambda_1se, newx = x_test)

# Calculate test MSE for Lasso and Ridge using both lambda.min and lambda.1se
lasso_mse_min <- mean((y_test - lasso_pred_min)^2)
lasso_mse_1se <- mean((y_test - lasso_pred_1se)^2)
```

```r
ridge_mse_min <- mean((y_test - ridge_pred_min)^2)
ridge_mse_1se <- mean((y_test - ridge_pred_1se)^2)

# Print the test MSE for both models
cat("Lasso Test MSE (lambda.min):", lasso_mse_min, "\n")
```

```
## Lasso Test MSE (lambda.min): 946414.7
```

```r
cat("Lasso Test MSE (lambda.1se):", lasso_mse_1se, "\n")
```

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

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

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

```r
cat("Ridge Test MSE (lambda.1se):", ridge_mse_1se, "\n")
```

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

Both Lasso and Ridge regression models were compared using cross-validation on the training data to select the optimal regularization parameter λ. The Ridge regression model with the optimal λ (`lambda.min`) achieved the lowest test Mean Squared Error (MSE) of 875,548.9, indicating a slightly better predictive performance compared to Lasso, which had a test MSE of 946,414.7 using the same λ. Ridge regression generally performed better in this case, especially with `lambda.min`, suggesting that shrinking the coefficients without setting them to zero (as Ridge does) may be more suitable for this dataset. Nonetheless, both models offer valid regularization techniques to improve prediction by controlling for multicollinearity and overfitting.

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.

In standard Lasso regression, the loss function is defined as:

$$L(\beta) = \frac{1}{2N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

where: - y is the vector of response variables, - X is the matrix of predictor variables, - β is the coefficient vector, - λ is the regularization parameter that controls the strength of the Lasso penalty, and - $\|\beta\|_1$ is the L1-norm (sum of absolute values of the coefficients).

The Lasso penalty forces some coefficients to be exactly zero, which leads to sparse models (selecting fewer features).

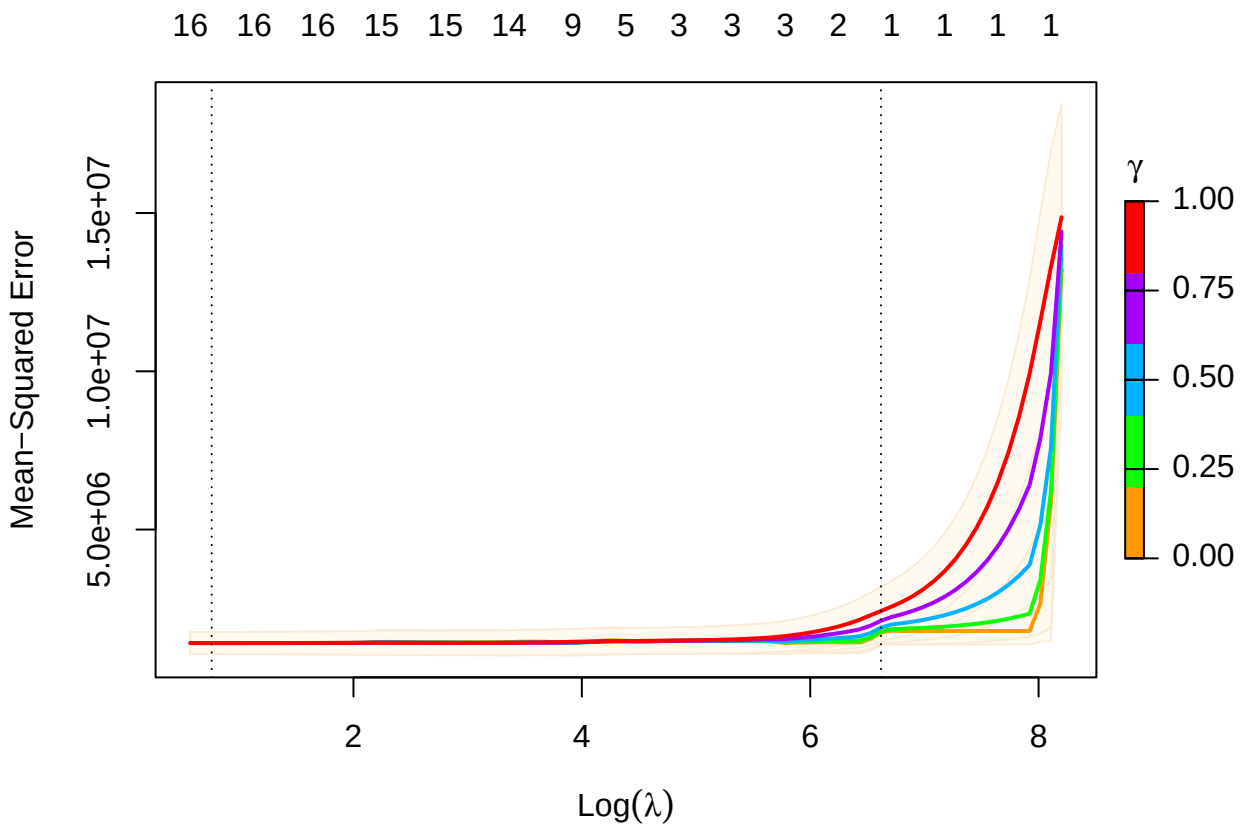
The `gamma` parameter in the `glmnet` package introduces a relaxation to the Lasso penalty by modifying the loss function as follows:

$$L_{relaxed}(\beta, \gamma) = \frac{1}{2N} \|y - X\beta_\lambda\|_2^2 + \gamma \left(\frac{1}{2N} \|y - X\beta_{OLS}\|_2^2 \right)$$

where: - β_λ is the coefficient vector from the Lasso fit with penalty λ, - β_{OLS} is the ordinary least squares (OLS) coefficient vector (i.e., the unregularized fit), - $\gamma \in [0, 1]$ is the blending parameter between the Lasso and OLS fits.

At $\gamma = 1$, the solution corresponds to the original Lasso fit. At $\gamma = 0$, the solution is closer to an OLS fit (less regularization). Thus, γ provides a way to relax the Lasso fit, resulting in a less sparse model.

```r
library(glmnet)

# Prepare the training and testing data as before
x_train <- model.matrix(Apps ~ ., data = train)[,-1]  # Remove the intercept column
y_train <- train$Apps

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

# Perform relaxed Lasso with cross-validation
set.seed(7)
cv_relaxed_lasso <- cv.glmnet(x_train, y_train, alpha = 1, relax = TRUE)

# Plot the cross-validation results for relaxed lasso
plot(cv_relaxed_lasso)
```




```
# Extract the optimal lambda and gamma values
optimal_lambda <- cv_relaxed_lasso$lambda.min
optimal_gamma <- cv_relaxed_lasso$relaxed$gamma.min

# Use the optimal lambda and gamma to make predictions on the test data
relaxed_lasso_pred <- predict(cv_relaxed_lasso, newx = x_test, s = "lambda.min", gamma = optimal_gamma)

# Calculate the test MSE for the relaxed Lasso model
relaxed_lasso_mse <- mean((y_test - relaxed_lasso_pred)^2)

# Print the results
cat("Relaxed Lasso Test MSE (optimal lambda and gamma):", relaxed_lasso_mse, "\n")
```

```
## Relaxed Lasso Test MSE (optimal lambda and gamma): 946414.7
```

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

```
## Optimal lambda: 2.134743
```

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

```
## Optimal gamma: 1
```

The relaxed Lasso model achieved a test MSE of 946,414.7 using the optimal λ value of 2.134743 and γ value of 1, which corresponds to the standard Lasso solution. This suggests that, in this case, relaxing the Lasso penalty did not significantly improve the performance over standard Lasso. The optimal γ being 1 implies that the full regularization provided by Lasso was sufficient, and no additional blending with the OLS solution was necessary to minimize the test error. Thus, the model performed well with strong regularization, indicating a sparse solution was appropriate for this dataset.

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, 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 <- as.matrix(t(golub))
set.seed(1)
y <- as.factor(golub.cl)


# Define a grid of alpha values from 0 to 1
alpha_grid <- seq(0, 1, by = 0.1)

# Use cross-validation to find the best alpha and lambda
cv_results <- list()
for (alpha_val in alpha_grid) {
  # Perform cross-validation for each alpha value
  cv_fit <- cv.glmnet(X, y, family = "binomial", alpha = alpha_val, nfolds = 10)
  cv_results[[as.character(alpha_val)]] <- cv_fit
}
```

```
## 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
# Extract the best model based on lambda.min
best_alpha <- 0
best_lambda <- 0
min_error <- Inf
for (alpha_val in alpha_grid) {
  cv_fit <- cv_results[[as.character(alpha_val)]]
  if (cv_fit$cvm[cv_fit$lambda == cv_fit$lambda.min] < min_error) {
    best_alpha <- alpha_val
    best_lambda <- cv_fit$lambda.min
    min_error <- cv_fit$cvm[cv_fit$lambda == cv_fit$lambda.min]
  }
}

cat("Best alpha:", best_alpha, "\n")
```

```
## Best alpha: 0.5
```

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

```
## Best lambda: 0.007829017
```

The logistic regression model using Elastic Net regularization selected an optimal α value of 0.5 and a λ value of 0.0078 through cross-validation. While these values provide a good fit for the data, caution should be taken due to the small sample size in one of the leukemia types, which could introduce some instability in the model's predictions.