

Stat 432 Homework 4

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

Contents

Question 1: Sparsity and Correlation	1
Question 2: Shrinkage Methods and Testing Error	6
Question 3: Penalized Logistic Regression	10

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:

$$Y = X^T \beta + \epsilon$$

Where $\beta = (\beta_1, \beta_2, \dots, \beta_{100})^T$ 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).

```
library(MASS)
library(glmnet)
```

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

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

```

# Step 1: Define the parameters
set.seed(1)
n_train <- 200
n_test <- 500
p <- 100
rho <- 0.1

# Step 2: Generate covariance matrix Sigma
Sigma <- matrix(rho, nrow = p, ncol = p)
diag(Sigma) <- 1

# Step 3: Generate training and test data
X_train <- mvrnorm(n_train, mu = rep(0, p), Sigma = Sigma)
X_test <- mvrnorm(n_test, mu = rep(0, p), Sigma = Sigma)

# True coefficient vector: only positions 1, 11, 21, 31 are nonzero
beta_true <- rep(0, p)
beta_true[c(1, 11, 21, 31)] <- 0.4

# Generate the response variable Y (training and test)
epsilon_train <- rnorm(n_train, mean = 0, sd = 1) # Random noise for training data
epsilon_test <- rnorm(n_test, mean = 0, sd = 1)   # Random noise for test data

Y_train <- X_train %*% beta_true + epsilon_train
Y_test <- X_test %*% beta_true + epsilon_test

# Step 4: Fit Lasso with 10-fold cross-validation using glmnet
lasso_fit <- cv.glmnet(X_train, Y_train, alpha = 1, nfolds = 10)

# Step 5: Select the optimal lambda using 'lambda.1se'
lambda_opt <- lasso_fit$lambda.1se

# Step 6: Make predictions on the test set
Y_pred <- predict(lasso_fit, newx = X_test, s = "lambda.1se")

# Step 7: Calculate the prediction error (Mean Squared Error)
mse <- mean((Y_test - Y_pred)^2)
print(paste("Test MSE: ", round(mse, 4)))

```

```
## [1] "Test MSE: 1.2898"
```

```

# Step 8: Check if the true model was selected
# The non-zero coefficients selected by Lasso
beta_lasso <- coef(lasso_fit, s = "lambda.1se")[-1] # Exclude the intercept
# Identify non-zero coefficients in Lasso
selected_vars <- which(beta_lasso != 0)
selected_vars

```

```
## [1] 1 11 21 31 92
```

```
true_vars <- c(1, 11, 21, 31)
```

```
# Report whether the true model was selected
true_model_selected <- all(selected_vars %in% c(1, 11, 21, 31)) && length(selected_vars) == 4
print(paste("True model selected: ", true_model_selected))
```

```
## [1] "True model selected: FALSE"
```

The MSE on the test data is 1.2898.

The true model was not 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.

```
n_train <- 200
n_test <- 500
p <- 100
rho_values <- c(0.1, 0.3, 0.5, 0.7, 0.9)
n_simulations <- 100

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

# Function to run a single simulation
run_simulation <- function(rho) {
  # Generate covariance matrix Sigma
  Sigma <- matrix(rho, nrow = p, ncol = p)
  diag(Sigma) <- 1 # Diagonal elements are 1

  # Generate training and test data
  X_train <- mvrnorm(n_train, mu = rep(0, p), Sigma = Sigma)
  X_test <- mvrnorm(n_test, mu = rep(0, p), Sigma = Sigma)

  # Generate the response variable Y
  epsilon_train <- rnorm(n_train) # Random noise for training data
  epsilon_test <- rnorm(n_test) # Random noise for test data
  Y_train <- X_train %*% beta_true + epsilon_train
  Y_test <- X_test %*% beta_true + epsilon_test

  # Fit Lasso with 10-fold cross-validation
  lasso_fit <- cv.glmnet(X_train, Y_train, alpha = 1, nfolds = 10)
  lambda_opt <- lasso_fit$lambda.1se

  # Make predictions on the test set
  Y_pred <- predict(lasso_fit, newx = X_test, s = "lambda.1se")

  # Calculate prediction error (MSE)
```

```

mse <- mean((Y_test - Y_pred)^2)

# Check if the true model was selected
beta_lasso <- coef(lasso_fit, s = "lambda.1se")[-1] # Exclude intercept
selected_vars <- which(beta_lasso != 0)
true_vars <- c(1, 11, 21, 31)
true_model_selected <- all(true_vars %in% selected_vars)

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

# Perform 100 simulation runs for each rho value and record results
results <- data.frame(rho = numeric(0), mse = numeric(0), model_selected = logical(0))

for (rho in rho_values) {
  for (i in 1:n_simulations) {
    sim_result <- run_simulation(rho)
    results <- rbind(results, data.frame(rho = rho, mse = sim_result$mse, model_selected = sim_result$model_selected))
  }
}

# Summarize the results: calculate average MSE and proportion of correct model selection for each rho
summary_results <- aggregate(cbind(mse, model_selected) ~ rho,
                             data = results, FUN = function(x) mean(x))
summary_results <- do.call(data.frame, summary_results)

# Print the summarized results
colnames(summary_results) <- c("rho", "avg_mse", "proportion_correct_model")
summary_results

##   rho  avg_mse proportion_correct_model
## 1 0.1 1.183328                0.97
## 2 0.3 1.196902                1.00
## 3 0.5 1.162357                0.96
## 4 0.7 1.180428                0.82
## 5 0.9 1.155993                0.04

```

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 4%, 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. The bias-variance trade-off becomes more pronounced as ρ increases. Higher correlation between covariates makes it more challenging for Lasso to correctly select the true variables and maintain prediction accuracy, illustrating the impact of multicollinearity on model performance. 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.

```

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

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

# Function to run a single simulation for Ridge Regression
run_ridge_simulation <- function(rho) {
  # Generate covariance matrix Sigma
  Sigma <- matrix(rho, nrow = p, ncol = p)
  diag(Sigma) <- 1 # Diagonal elements are 1

  # Generate training and test data
  X_train <- mvrnorm(n_train, mu = rep(0, p), Sigma = Sigma)
  X_test  <- mvrnorm(n_test, mu = rep(0, p), Sigma = Sigma)

  # Generate the response variable Y
  epsilon_train <- rnorm(n_train) # Random noise for training data
  epsilon_test  <- rnorm(n_test)  # Random noise for test data
  Y_train <- X_train %*% beta_true + epsilon_train
  Y_test  <- X_test  %*% beta_true + epsilon_test

  # Fit Ridge with 10-fold cross-validation
  ridge_fit <- cv.glmnet(X_train, Y_train, alpha = 0, nfolds = 10)
  lambda_opt <- ridge_fit$lambda.1se

  # Make predictions on the test set
  Y_pred <- predict(ridge_fit, newx = X_test, s = "lambda.1se")

  # Calculate prediction error (MSE)
  mse <- mean((Y_test - Y_pred)^2)

  return(mse)
}

# Store results in a list instead of appending to a data frame
ridge_results <- list()

# Perform 100 simulation runs for each rho value and record results
for (rho in rho_values) {
  rho_ridge_results <- vector("list", n_simulations) # Store results for each simulation

  for (i in 1:n_simulations) {
    rho_ridge_results[[i]] <- c(rho = rho, mse = run_ridge_simulation(rho))
  }
}

```

```

}

# Combine results for this rho
ridge_results <- c(ridge_results, rho_ridge_results)
}

# Convert the list to a data frame
ridge_results_df <- do.call(rbind, lapply(ridge_results, function(x) data.frame(rho = x[["rho"]], mse =

# Summarize the results: calculate average MSE for each rho
ridge_summary_results <- aggregate(mse ~ rho, data = ridge_results_df, FUN = mean)
ridge_summary_results

##   rho      mse
## 1 0.1 1.460023
## 2 0.3 1.403247
## 3 0.5 1.349703
## 4 0.7 1.268042
## 5 0.9 1.148494

```

The average prediction error (MSE) decreases as the correlation ρ between predictors increases. For $\rho = 0.1$, the average MSE is 1.46, while for $\rho = 0.9$, the average MSE drops to 1.14. This 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. Comparing Ridge with Lasso, a key difference is 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 does not perform variable selection but shrinks all coefficients. As a result, Ridge Regression typically performs better when many variables contribute to the prediction, especially in cases of high correlation ρ (multicollinearity), as indicated by the decreasing MSE.

In contrast, Lasso performs variable selection, and its prediction accuracy is influenced by both the degree of correlation and sparsity of the data. 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.

Key Insights:

- Ridge may show better performance (lower MSE) than Lasso when the covariates are highly correlated, as Lasso tends to struggle with multicollinearity.
- Bias-Variance Tradeoff: Ridge introduces more bias but reduces variance, which can result in more stable predictions when the predictors are correlated.
- In sparse settings (when only a few variables have non-zero coefficients), Lasso may outperform Ridge by removing irrelevant variables, but Ridge generally handles high correlation better across all variables.

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,]
```

```
# Fit the linear model on the training set
```

```
linear_model <- lm(Apps ~ ., data = train)
```

```
summary(linear_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
```

```
# Predict the number of applications on the test set
predictions <- predict(linear_model, test)
```

```
# Calculate the test error (MSE)
test_mse <- mean((predictions - test$Apps)^2)
print(paste("Test MSE: ", test_mse))
```

```
## [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.

```
# Prepare the training and test sets
```

```
x_train <- model.matrix(Apps ~ ., train)[-1] # Remove the intercept column
y_train <- train$Apps
```

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

```
# Ridge Regression using cross-validation
```

```
set.seed(7)
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0)
```

```
# Optimal lambda values for Ridge
```

```
lambda_min_ridge <- cv_ridge$lambda.min # Lambda that minimizes the cross-validation error
lambda_1se_ridge <- cv_ridge$lambda.1se # Largest lambda within 1 standard error of the minimum
```

```
# Predict on the test set using the two optimal lambdas
```

```
ridge_pred_min <- predict(cv_ridge, s = lambda_min_ridge, newx = x_test)
ridge_pred_1se <- predict(cv_ridge, s = lambda_1se_ridge, newx = x_test)
```

```
# Calculate the test MSE for Ridge regression
```

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

```
# Lasso Regression using cross-validation
```

```
set.seed(7)
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1)
```

```
# Optimal lambda values for Lasso
```

```
lambda_min_lasso <- cv_lasso$lambda.min
lambda_1se_lasso <- cv_lasso$lambda.1se
```



```

# Predict on the test set using the two optimal lambdas
lasso_pred_min <- predict(cv_lasso, s = lambda_min_lasso, newx = x_test)
lasso_pred_1se <- predict(cv_lasso, s = lambda_1se_lasso, newx = x_test)

# Calculate the test MSE for Lasso regression
lasso_mse_min <- mean((lasso_pred_min - y_test)^2)
lasso_mse_1se <- mean((lasso_pred_1se - y_test)^2)

cat("Ridge Regression Test MSE (lambda.min):", round(ridge_mse_min, 2), "\n")

```

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

```
cat("Ridge Regression Test MSE (lambda.1se):", round(ridge_mse_1se, 2), "\n")
```

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

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

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

```
cat("Lasso Regression Test MSE (lambda.1se):", round(lasso_mse_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 explanation 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.

Answer:

The relaxed fit in Lasso regression adds flexibility to the model by introducing a new tuning parameter γ , which determines how much the Lasso solution is “shrunk” towards a simpler model. Essentially, γ controls how much we relax the Lasso solution, allowing us to find a balance between a fully constrained model (Lasso) and an unconstrained model (OLS). To explain the relaxed fit, let’s first define the main components:

Let $y \in \mathbb{R}^n$ be the response vector representing the number of applications (Apps), and $X \in \mathbb{R}^{n \times p}$ be the matrix of predictors (the other variables).

Lasso regression minimizes the following loss function:

$$\hat{\beta}_\lambda = \arg \min_{\beta} \left\{ \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

where λ controls the strength of the Lasso penalty, which encourages sparsity by shrinking some coefficients to zero. However, too much shrinkage can over-regularize the model by removing useful variables.

The relaxed fit introduces a new tuning parameter, γ , that combines the Lasso solution with an ordinary least squares (OLS) refit on the selected variables. Mathematically, it works as follows:

First, fit the Lasso model to get an estimate $\hat{\beta}_\lambda$.

Then, blend this solution with an OLS fit on the selected variables:

$$\hat{\beta}_{\lambda,\gamma} = \gamma \cdot \hat{\beta}_\lambda + (1 - \gamma) \cdot \hat{\beta}_{OLS}$$

where $\hat{\beta}_{OLS}$ is the OLS estimate and $\gamma \in [0, 1]$ controls the mix between Lasso ($\gamma = 1$) and OLS ($\gamma = 0$).

The goal is to reduce over-regularization by allowing the non-zero Lasso coefficients to “relax” and be refitted with less or no penalty, improving model performance when certain variables shouldn’t be fully shrunk to zero.

```
# Prepare the training and test sets
x_train <- model.matrix(Apps ~ ., train)[-1] # Remove the intercept column
y_train <- train$Apps

x_test <- model.matrix(Apps ~ ., test)[-1]
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)

# Extract optimal lambda and gamma
lambda_min_relaxed <- cv_relaxed_lasso$lambda.min
gamma_min_relaxed <- cv_relaxed_lasso$relaxed$gamma.min

# Predict on the test set using the optimal lambda and gamma
relaxed_pred <- predict(cv_relaxed_lasso, s = lambda_min_relaxed, gamma = gamma_min_relaxed, newx = x_test)

# Calculate the test MSE for relaxed Lasso
relaxed_mse <- mean((relaxed_pred - y_test)^2)

cat("Test MSE for the relaxed Lasso model:", relaxed_mse, "\n")
```

```
## Test MSE for the relaxed Lasso model: 946414.7
```

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 875548.9, indicating a slightly better predictive performance compared to Lasso, which had a test MSE of 946414.7 using the same λ . Ridge regression generally performed better in this case, especially with `lambda.min`. This shows 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.

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")'.
```

```
library(glmnet)
```

```
data(golub)
```

```
# The outcome variable (golub.cl) is the leukemia type (ALL or AML)
```

```
y <- golub.cl
```

```
# The predictor variables are the gene expressions (3051 genes)
```

```
X <- t(golub) # Transpose to get samples as rows and genes as columns
```

```
# Convert the outcome variable to a binary factor
```

```
y <- as.factor(y)
```

```

# Set a sequence of alpha values for Elastic Net
alpha_values <- seq(0, 1, by = 0.1)

# Perform cross-validation for each alpha value
cv_results <- lapply(alpha_values, function(alpha) {
  cv.glmnet(X, y, family = "binomial", alpha = alpha)
})

# Extract the best models from the cross-validation results
cv_mins <- sapply(cv_results, function(cv) cv$cvm[cv$lambda == cv$lambda.min])
best_alpha_index <- which.min(cv_mins)
best_alpha <- alpha_values[best_alpha_index]
best_model <- cv_results[[best_alpha_index]]

# Get the best lambda from the selected model
best_lambda <- best_model$lambda.min

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

```

```
## Best alpha: 0.3
```

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

```
## Best lambda: 0.01304836
```