Project Report: Project 1 LASSO vs OLS

MAT 4376

Victor Matyiku, Rebecca Ly

2025-03-30

Contents

1.Introduction

1.1 Background of OLS and LASSO Method

OLS (Ordinary Least Squares) and **LASSO** (Least Absolute Selection and Shrinkage Operator) are both methods that can be used to estimate regression coefficients for a line of best fit on data. These coefficients are typically represented by β, a p-dimensional vector where p is the number of predictors.

The objective of the OLS method is to find a line of best fit that minimizes the sum of squared residuals, which are the differences between the predicted and actual data. This method is effective when the sample size (n) of the chosen dataset is greater than the number of predictors (p), i.e $n > p$. Otherwise the method is undefined. The OLS estimator derived from the method can be written as follows:

$$\hat{\beta}_{OLS} = argmin_{\beta} \frac{1}{2n} \|Y - X\beta\|_2^2$$

where

- $Y = (Y_1, ..., Y_n)^T$ is a $n \times 1$ response vector
- $X = (X_1, ..., X_n)^T$ is a $n \times p$ design matrix of covariates
- $\beta = (Y_1, ..., Y_n)^T$ is a p-dimensional regression coefficient vector

For $p > n$, the LASSO method is more favourable. The LASSO method uses L1 regularization where a penalty is added to the structure of the OLS method. The penalty achieves sparsity: an effect where the coefficients of some features are reduced close to, or exactly to zero, resulting in only certain features being selected in the final model. In other words, predictors with the least predictive power are dropped from the model and the remaining are seen as the most significant predictors. The estimator is typically written in Lagrangian form as follows:

$$\hat{\beta}_{LASSO} = argmin_{\beta}\{L(\beta) + \lambda\|\beta\|_1\}$$

$$\hat{\beta}_{LASSO} = argmin_{\beta}\{\frac{1}{2n}\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\} = argmin_{\beta}F(\beta)$$

One way of tuning the parameter lambda from a set of candidates is through *Cross-Validation (CV)*. The candidate that produces the LASSO estimator that minimizes the loss function is selected to be the most optimal lambda. A detailed procedure of the CV method is listed below:

1. Divide the data set n into m disjoint set $D_1, ..., D_m$ of size n/m each.
2. For each $\lambda \in \Lambda$, evaluate $\hat{\beta}^{-(h)}_{LASSO}(\lambda)$. the LASSO estimator based on the dataset $D \setminus D_h$, $h = 1, ..., m$. Each D_h is treated as a *test dataset*, while $D \setminus D_h$ as a *training dataset*.
3. Thus, for each $\lambda \in \Lambda$ we get h LASSO estimators, making in total $h \times q$ LASSO estimators.
4. Define the loss function
$$CV(\lambda) = \sum_{h=1}^{m} \sum_{i:(X_i,Y_i)\in D_h} (Y_i - X_i^T \hat{\beta}^{-(h)}_{LASSO}(\lambda))^2$$
5. Choose λ that minimized the loss function.

It can be seen that the OLS estimator can be derived from the LASSO estimator as $t \to \infty$. It should also be noted that while LASSO can set coefficients to zero, there is the risk of over-shrinkage.

1.2 Hypothesis Statement

- LASSO is expected to show better results [finish]

1.3 Goals

1. Write a function for OLS using `lm()`
2. Write a function for LASSO using λ chosen from cross-validation
3. Successfully implement one hundred iterations of both functions for each chosen value of p (500 iterations total)
4. Summarize the number of chosen predictors for each p and record the mean squared value

2. Data source and Design

2.1 Data Source

The program, designed in R, used five values of $p = \{2, 5, 10, 15, 25\}$. Each iteration on p created a randomly generated, p-dimensional dataset based on the following linear model:

$$Y = \beta_0 + X\beta + \epsilon$$

where

- $Y = (Y_1, ..., Y_n)^T$ is a $n \times 1$ response vector
- $X = (X_1, ..., X_n)^T$ is a $n \times p$ design matrix of covariates
- $\beta = (Y_1, ..., Y_n)^T$ is a p-dimensional regression coefficient vector
- β_0 is the intercept
- ϵ is the n-dimensional noise vector

No other manipulation was performed on the dataset.

A total of 3 functions were written for this project: `generate_data()`, `ols()`, and `lasso()`. `generate_data()` initially creates the normally distributed data set X using the number of predictors (p) and number of observations (n).

2.2 Design of OLS Function

- describe functions [?]

2.3 Design of LASSO Function

- describe functons [?]

3. Results

4. Discussion

5. Appendix

Code

Generating Data

```r
generate_data <- function(p, n){
  beta0 <- rnorm(1)
  beta <- matrix(rnorm(p), nrow = p, ncol = 1)

  predictors <- matrix(
    rnorm(n*p), nrow = n, ncol = p,
    dimnames = list(rows = 1:n, cols = paste("X", 1:p, sep=""))
  )

  errors <- rnorm(n)
  observations <- data.frame(Y = beta0 + predictors %*% beta + errors)

  output <- as.data.frame(
    cbind(observations, predictors)
  )

  return(output)
}

#generated_data <- generate_data(10, 100)
#head(generated_data)
```

OLS

```r
ols <- function(input_data){
  ols_model <- lm(Y ~ ., input_data)

  # extract only coefficients whose p value <= 0.5
  significant_coefs <- data.frame(Coefficients = summary(ols_model)$coef[summary(ols_model)$coef[,4] <=

  # transpose vector so calculations work properly later
  ols_coefs <- t(significant_coefs)
  significant_predictors <- colnames(ols_coefs)[colnames(ols_coefs) != "(Intercept)"]
  ols_coefs <- ols_coefs[, significant_predictors]

  # check if intercept is significant
  if (is.na(significant_coefs["(Intercept)",])) {
   ols_beta_0 <- 0
 } else {
   ols_beta_0 <- significant_coefs["(Intercept)",]
 }

  sig_predictors_matrix <- as.matrix(input_data[, significant_predictors])
  errors <- input_data$Y - ols_beta_0 - sig_predictors_matrix %*% ols_coefs

  # calculating squared error using euclidean norm
  output <- list(sq_error = (norm(errors, "e"))^2,
                 q = length(significant_predictors))

  return(output)
}
```

```r
#ols(generated_data)
```

LASSO

```r
library(glmnet)
```

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

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

```r
lasso <- function(input_data){
  X <- as.matrix(input_data[colnames(input_data) != "Y"])
  Y <- as.matrix(input_data$Y)

  # LASSO with CV
  cv.fit <- cv.glmnet(X, Y, nfolds=10, family="gaussian")

  # coefficients for LASSO model using optimal lambda
  lasso_coefs <- coef(cv.fit, s="lambda.min")

  no_intercept <- lasso_coefs@i[lasso_coefs@i != 0]

  #predict new coefficients using LASSO & optimal lambda
  prediction <- predict(cv.fit, newx = X, s="lambda.min")

  # calculating squared error using euclidean norm
  output <- list(sq_error = (norm(Y-prediction, "e"))^2,
                 q = length(no_intercept))

  return(output)
}

#lasso(generated_data)
```

Repetition

```r
analysis_for <- function(p) {
  statistics <- list(
    ols_err = 1:100,
    lasso_err = 1:100,
    ols_q = 1:100,
    lasso_q = 1:100
  )

  for (i in 1:100) {
    generated_data <- generate_data(p, 100)
    ols_stats <- ols(generated_data)
    lasso_stats <- lasso(generated_data)
```

```r
    statistics$ols_err[i] <- ols_stats$sq_error
    statistics$lasso_err[i] <- lasso_stats$sq_error
    statistics$ols_q[i] <- ols_stats$q
    statistics$lasso_q[i] <- lasso_stats$q
  }

  statistics$ols_mse <- sum(statistics$ols_err)/100
  statistics$lasso_mse <- sum(statistics$lasso_err)/100

  return(statistics)
}

#analysis_for(10)
```

P values

```r
p_values <- c(2, 5, 10, 25, 50)

summary_df <- data.frame(
  p = p_values,
  OLS = p_values,
  LASSO = p_values #just to be consistent in size
)

for (i in 1:5) {
  p <- p_values[i]

  #getting statistics
  statistics <- analysis_for(p)
  ols_mse <- statistics$ols_mse
  lasso_mse <- statistics$lasso_mse

  #inputting into matrix
  summary_df$OLS[i] <- ols_mse
  summary_df$LASSO[i] <- lasso_mse

  print(paste0("Summary Statistics for p = ", p))
  writeLines("\nOLS Statistics\n")
  print(summary(statistics$ols_q))
  writeLines("\nLASSO Statistics\n")
  print(summary(statistics$lasso_q))
  writeLines("\n")
}
```

```
## [1] "Summary Statistics for p = 2"
##
## OLS Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    2.00    1.66    2.00    2.00
##
```

```
## LASSO Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    2.00    2.00    1.83    2.00    2.00
##
##
## [1] "Summary Statistics for p = 5"
##
## OLS Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    4.00    4.00    4.22    5.00    5.00
##
## LASSO Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00    5.00    5.00    4.79    5.00    5.00
##
##
## [1] "Summary Statistics for p = 10"
##
## OLS Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     5.0     8.0     9.0     8.5     9.0    10.0
##
## LASSO Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.00   10.00   10.00    9.81   10.00   10.00
##
##
## [1] "Summary Statistics for p = 25"
##
## OLS Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.00   19.00   20.50   20.37   21.25   24.00
##
## LASSO Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.00   24.00   25.00   24.53   25.00   25.00
##
##
## [1] "Summary Statistics for p = 50"
##
## OLS Statistics
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   33.00   37.00   39.00   39.29   41.00   47.00
##
## LASSO Statistics
##
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   45.00   49.00   49.00   49.17   50.00   50.00
```

```r
writeLines("Summary matrix of LASSO and OLS MSE\n")
```

```
## Summary matrix of LASSO and OLS MSE
```

```r
kable(summary_df)
```

p	OLS	LASSO
2	383.0045	97.75122
5	770.3763	92.73462
10	1219.9933	89.09582
25	2646.3632	74.15521
50	5178.7248	49.46761