STAT 154 Lab 5: Ordinary Least Squares (OLS)

Yuansi Chen and Raaz Dwivedi

March 4, 2019

The goal of this lab is to review some theoretical properties and computational aspects of ordinary least squares (OLS) regression. Questions with prefix (a), (b), ... are to be answered by you.

1 Ordinary least squares

We observe n i.i.d. samples of response-predictor pair $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$. For convenience purpose, we assume that $n > d$. (The next lecture will deal with the case $n \leq d$.)

The ordinary least squares solves β by minimizing the following objective

$$\min_{\beta \in \mathbb{R}^d} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 , \tag{1}$$

where $\mathbf{y}$ is an n-vector of responses values y_i, $\mathbf{X}$ is the $n \times d$ matrix of predictors with each row equals to $x_i^\top$.

$$\mathbf{X} = \begin{pmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_n^\top \end{pmatrix}$$

To include the intercept term, we assume that the first column of $\mathbf{X}$ is a vector of ones. The resulting vector

$$\hat{\beta} \in \arg\min_{\beta \in \mathbb{R}^d} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 ,$$

is the regression coefficient vector.

There are at least two ways to solve a minimization problem like equation (1). Denote

$$f(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|_2^2 .$$

1. Set gradient to zero and look for a closed form solution. (A closed form solution does not always exist.) Solve

$$\nabla f(\beta) = 0.$$

2. Use an iterative optimization algorithm like gradient descent. (Almost always doable as long as we can compute gradient, but not alwasy guaranteed to converges to the gradient-zero solution.)

$$\beta_{k+1} = \beta_k - \eta \nabla f(\beta_k).$$

For OLS, if $\mathbf{X}^\top \mathbf{X}$ is invertible, we have a closed form solution

$$\hat{\beta} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top y. \tag{2}$$

1.1 When do we have the closed-form solution?

(a) When is $\mathbf{X}^\top\mathbf{X}$ is non-invertible? Let $\mathbf{X}_{\cdot j}$ be the j-th column of $\mathbf{X}$ or j-th feature. What happens when two features are identical $\mathbf{X}_{\cdot j} = \mathbf{X}_{\cdot k}$? What happens when $\mathbf{X}_{\cdot j} = \mathbf{X}_{\cdot k} + \mathbf{X}_{\cdot l}$?

(b) Show that when $n < d$, $\mathbf{X}^\top\mathbf{X}$ is non-invertible.

(c) Generate a random matrix $\mathbf{X}$ of size $n \times d$. $n = 100$, $d = 10$. Compute the eigenvalues of $\mathbf{X}^\top\mathbf{X}$. Repeat this for 1000 times and also with different choices of n and d. Show in simulation that for a random matrix $\mathbf{X}$, as long as $n > d$, $\mathbf{X}^\top\mathbf{X}$ is almost always invertible.

1.2 Gradient method on OLS

Briefly review the gradient method on OLS we discussed in class.

1.3 Common outputs of OLS

OLS procedure typically outputs the following quantities

- Regression cofficient estimates: $\hat{\beta} = \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_d\right)^\top$.
- Predcited (or fitted) response values: $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)^\top$.

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$$

- Residuals: $e_i = y_i - \hat{y}_i$.

The secondary results are directly derived from previous outputs.

- Residual Sum of Squares (RSS):

$$RSS = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

- Coefficient of determination R^2:

$$R^2 = 1 - \frac{RSS}{TSS},$$

where TSS is the total sum of squares:

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2.$$

(a) Show that the coefficient of determination equals to the squared correlation between the observed responses and the predicted responses. That is

$$R^2 = \mathrm{cor}^2(\mathbf{y}, \hat{\mathbf{y}})$$

(b) What does it mean to have R^2 close to 1?

2 Linear Model

Review the lecture slides and recall the difference between the least squares and the linear model.

In the linear model, given $x_i \in \mathbb{R}^d$, we assume that each response y_i is generated i.i.d. from the following model

$$y = x^\top \beta^* + \epsilon, \tag{3}$$

where the error ϵ_i is generated i.i.d. with mean zero and constant variance σ^2. Using matrix notation, we can express the above model as

$$\mathbf{y} = \mathbf{X}\beta^* + \boldsymbol{\epsilon}, \tag{4}$$

where $\boldsymbol{\epsilon}$ is an n-vector of error values. Note that here we made strong assumption that there exists a "true" $\beta^* \in \mathbb{R}^d$ that generates all the responses.

Under the linear model,

(a) show that the OLS estimates $\hat{\beta}$ is an unbiased estimator of β^*. That is

$$\mathbb{E}\left[\hat{\beta}\right] = \beta^*.$$

(b) show that the following variance estimator

$$\hat{\sigma}^2 = \frac{RSS}{n-d},$$

is an unbiased estimator of the error variance σ^2.

3 OLS in action

We are going to use OLS on the Auto MPG Data Set (See UCI repository). Download the cleaned version of the data set on piazza (mpg.csv).

```
# set working directory
setwd("/Users/yuansichen/UCB/Teaching/2019_Spring/Problems/stat154copy/labs/lab5/")
# read dataset
data <- read.csv("mpg.csv")
print(head(data))
```

```
##   mpg cylinders displacement horsepower weight acceleration model_year
## 1  18         8          307        130   3504         12.0         70
## 2  15         8          350        165   3693         11.5         70
## 3  18         8          318        150   3436         11.0         70
## 4  16         8          304        150   3433         12.0         70
## 5  17         8          302        140   3449         10.5         70
## 6  15         8          429        198   4341         10.0         70
##   origin                       name
## 1      1 chevrolet chevelle malibu
## 2      1         buick skylark 320
## 3      1        plymouth satellite
## 4      1             amc rebel sst
## 5      1               ford torino
## 6      1          ford galaxie 500
```

```
cat("nrows = ", nrow(data), "ncols = ", ncol(data), "\n")
```

```
## nrows =  398 ncols =  9
```

To simplify the problem, we will compute OLS to predict **mpg** using three other variables.

$$\textbf{mpg} \approx \beta_1 \textbf{horsepower} + \beta_2 \textbf{weight} + \beta_3 \textbf{acceleration} + \beta_4,$$

First, we leave 10% of the dataset for test purpose. Make sure you only touch the test data once. The rest data can be used in multiple ways. You can either use the entire rest data for training, or split the rest data into training and validation two parts.

```
set.seed(123456)
test_size <- floor(nrow(data)*0.1)
test_ind <- sample(seq_len(nrow(data)), size = test_size)
dataTest <- data[test_ind, ]
dataUse <- data[-test_ind, ]
cat("nrows = ", nrow(dataUse), "ncols = ", ncol(dataUse), "\n")

## nrows =  359 ncols =  9
```

3.1 Exploratory Data Analysis

Before running OLS, it is always helpful to spend some time exploring the variables above.

- Summary statistics
- Histograms and boxplots for each variable
- correlation between variables (both predictors and responses)
- scatterplot
- PCA, K-means etc.

3.2 Write your own OLS function

Your mission here is to write your own R code to run OLS without using **lm()** function. Ideally, you should try to encapsulate your code using a function rather than a couple of lines. Inputs:

- predictors $\mathbf{X}$: $n \times d$ matrix
- responses $\mathbf{y}$: n-vector

Outputs:

- **coefficients**: vector of regression coefficient estimates $\hat{\beta}$
- **fitted_values**: vector of fitted values $\hat{\mathbf{y}}$
- **residuals**: vector of residuals

Some useful functions in R:

- **cbind()**: column binding
- **solve()**: solve a linear system or compute the inverse of a matrix.

```
myOLS <- function(X, Y){
  # Compute oridinary least squares

  # Args:
  #   X: predictor matrix, obs * variables
  #   Y: responses
  #
  # Returns:
  #   coefficients: vector of regression coefficient estimates
  #   fitted_values: vector of fitted values
  #   residuals: vector of residuals

  # TODO
  output_list <- list("coefficients" = NULL, "fitted_values" = NULL, "residuals" = NULL)
  return(output_list)
  }
```

Compare your results with **lm()**. Note that **lm()** automatically adds a column of ones to account for intercept.

```
# regression output with lm()
res_OLS <- lm(mpg ~ as.numeric(horsepower) + weight + acceleration, data = dataUse)
res_OLS$coefficients
```

```
##            (Intercept) as.numeric(horsepower)                  weight
##           40.088180708            0.006111277           -0.007081530
##           acceleration
##            0.259796876
```

(a) try **summary()** on the output of **lm()**.

(b) try **plot()** on the output of **lm()**.

(c) Given your estimtor $\hat{\beta}$, compute the mean squared error (MSE) on the test data

(d) Computer the MSE on the test data with that on the training data.

3.3 Visualizing the residuals

Visualizing the residuals is one way to check whether the linear model assumption is valid.

(a) Plot the residuals as a function of **acceleration**. Do you believe the assumption that the residual is independent of **acceleration**?

(b) Plot the residuals and fitted values in the same figure. Here you just need to create a new data

```
library(ggplot2)
# TODO, replace the two 0s with your results.
postAnalysis <- cbind(dataUse, fitted.values=0, residuals=0)
# Plot, no need to change the following lines
ggplot(postAnalysis, aes(x = acceleration, y = mpg)) +
  geom_smooth(method = "lm", se = FALSE, color = "lightgrey") +
```

```
geom_segment(aes(xend = acceleration, yend = fitted.values), alpha = .2) +

# > Color AND size adjustments made here...
geom_point(aes(color = abs(residuals), size = abs(residuals))) + # size also mapped
scale_color_continuous(low = "black", high = "red") +
guides(color = FALSE, size = FALSE) +  # Size legend also removed
# <

geom_point(aes(y = fitted.values), shape = 1) +
theme_bw()
```

3.4 Data transformation

Here we have first used OLS with "raw" data. However, in practice, we often want to fit OLS with transformed data. Here we discuss two popular transformations: mean-centered data, and standardized data.

Mean-centered data

(a) Use **lm()** or your implementation to recompute OLS coefficients $\hat{\beta}$ by regressing untransformed **mpg** on mean centered predictors **horsepower**, **weight**, **acceleration**.

(b) With the new estimator, how do you recover the previous estimator from untransformed variables?

Standardized predictors

(a) Use **lm()** or your implementation to recompute OLS coefficients $\hat{\beta}$ by regressing untransformed **mpg** on standardized predictors **horsepower**, **weight**, **acceleration**.

(b) With the new estimator, how do you recover the previous estimator from untransformed variables?

3.5 Handling Catergoiral Variables

Not all predictors are continuous variables or quantitative variables. Sometimes there will be categorial variables. In OLS regression, one typical treatment for a categorial variable is to transform it into a set of dummy variables. For example, for a categorial variable x with $k = 3$ categories a, b and c. We can dummify it by creating $k - 1 = 2$ dummy indicators or k dummy indicators.

$$x = \begin{pmatrix} a \\ b \\ c \\ b \\ a \end{pmatrix} \Rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ or } \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

(a) Consider the variable **origin**. This is a categorial one: 1=USA, 2=Europe, 3=Japan. Form a new design matrix by adding two dummy indicators for **origin**.

(b) Using your OLS implementation to compute new coefficient estimates.

(c) Compare it with the following call of **lm()**. Which dummification does R use?

```r
# regression output with lm()
res_OLS2 <- lm(mpg ~ as.numeric(horsepower) + weight + acceleration + factor(origin), data = dataUse)
res_OLS2$coefficients
```

```
##            (Intercept) as.numeric(horsepower)                  weight
##           37.876981976            0.004853578           -0.006578707
##           acceleration        factor(origin)2        factor(origin)3
##            0.277607413            0.437347388            2.143486751
```