Problem Set 1

Yuanyuan Liu

Due: February 11, 2024

Instructions

- Please show your work! You may lose points by simply writing in the answer. If the problem requires you to execute commands in `R`, please include the code you used to get your answers. Please also include the `.R` file that contains your code. If you are not sure if work needs to be shown for a particular problem, please ask.
- Your homework should be submitted electronically on GitHub in `.pdf` form.
- This problem set is due before 23:59 on Sunday February 11, 2024. No late assignments will be accepted.

Question 1

The Kolmogorov-Smirnov test uses cumulative distribution statistics test the similarity of the empirical distribution of some observed data and a specified PDF, and serves as a goodness of fit test. The test statistic is created by:

$$D = \max_{i=1:n} \left\{ \frac{i}{n} - F_{(i)}, F_{(i)} - \frac{i-1}{n} \right\}$$

where F is the theoretical cumulative distribution of the distribution being tested and $F_{(i)}$ is the ith ordered value. Intuitively, the statistic takes the largest absolute difference between the two distribution functions across all x values. Large values indicate dissimilarity and the rejection of the hypothesis that the empirical distribution matches the queried theoretical distribution. The p-value is calculated from the Kolmogorov- Smirnoff CDF:

$$p(D \leq x) \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} e^{-(2k-1)^2\pi^2/(8x^2)}$$

which generally requires approximation methods (see Marsaglia, Tsang, and Wang 2003). This so-called non-parametric test (this label comes from the fact that the distribution of the test statistic does not depend on the distribution of the data being tested) performs poorly in small samples, but works well in a simulation environment. Write an `R` function that implements this test where the reference distribution is normal. Using `R` generate 1,000 Cauchy random variables (`rcauchy(1000, location = 0, scale = 1)`) and perform the test (remember, use the same seed, something like `set.seed(123)`, whenever you're generating your own data).

As a hint, you can create the empirical distribution and theoretical CDF using this code:

```r
# create empirical distribution of observed data
ECDF <- ecdf(data)
empiricalCDF <- ECDF(data)
# generate test statistic
D <- max(abs(empiricalCDF - pnorm(data)))
```

```r
# Function to perform the KS test
ks_test <- function(data) {
  n <- length(data)
  # Sort data for ECDF calculation
  data <- sort(data)
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)

  # Theoretical CDF for the standard normal distribution
  theoreticalCDF <- pnorm(data)


  # Function to calculate the p-value using matrix H
  ks_pvalue_matrix <- function(D, n) {
    k <- ceiling(n * D)
    h <- k - n * D
    m <- 2 * k - 1

    # Fill the matrix
    # Initialize the matrix H
    H <- matrix(0, nrow = m, ncol = m)

    for (i in 1:m) {
      for (j in 2:i) {
        H[i, j] <- (1-h^(i+1-j)) / factorial(i+1-j)
      }
    }

    for (i in 1:(m-1)) {
      for (j in 1) {
        H[i, j] <- (1-h^i) / factorial(i)
      }
    }

    for (i in 1:m) {
      for (j in 1) {
        H[i, j] <- (1-2*h^i) / factorial(i)
```

```r
      }
    }

    for (i in 2:(m-1)) {
      for (j in 1:i) {
        if (i+1 <= j) {
          H[i, j] <- 1 / factorial(i+1-j)
        } else {
          H[i, j] <- 0
        }
      }
    }

    T <- H
    for (i in 2:n) {
      T <- T %*% H
    }
    p_value <- factorial(n) * T[k, k] / n^n
    return(p_value)
  }
  D <- max(abs(empiricalCDF - theoreticalCDF))
  p_value <- ks_pvalue_matrix(D, n)

  return(list(D = D, p_value = p_value))
}

# Set seed for reproducibility
set.seed(123)

# Generate 1,000 Cauchy random variables
cauchy_data <- rcauchy(1000, location = 0, scale = 1)

# Perform the KS test
ks_result <- ks_test(cauchy_data)

# Print the result
print(ks_result)
```

The result is:
D:0.1347281
p_value:NaN

Question 2

Estimate an OLS regression in `R` that uses the Newton-Raphson algorithm (specifically `BFGS`, which is a quasi-Newton method), and show that you get the equivalent results to using `lm`. Use the code below to create your data.

```r

  # Function to calculate the p-value using matrix H
  ks_pvalue_matrix <- function(D, n) {
```

```r
# Set the seed for reproducibility
set.seed(123)

# Create the data
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75 * data$x + rnorm(200, 0, 1.5)

# Estimate the OLS regression using lm()
lm_fit <- lm(y ~ x, data = data)

# Now using optim() to perform OLS manually using the BFGS method
# Define the objective function (sum of squared residuals)
ssr <- function(params, data) {
  with(data, sum((y - (params[1] + params[2] * x))^2))
}

# Initial parameter guesses
initial_params <- c(0, 0)

# Run optim() with BFGS method
optim_fit <- optim(par = initial_params, fn = ssr, data = data, method = "BFGS
    ")

# Show the coefficients from lm and optim
lm_coefficients <- coef(lm_fit)
optim_coefficients <- optim_fit$par
# Print the results
print(lm_coefficients)
print(optim_coefficients)
```

Estimate an OLS regression using BFGS:
(Intercept) x
0.1391778 2.7267000
Estimate an OLS regression using lm:
(Intercept) x
0.1391874 2.7266985