Problem Set 1

Applied Stats II

Due: February 12, 2023

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 12, 2023. 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
#Heres an R function that implements the Kolmogorov-Smirnov test for the normal distribution:

kolmogorov.smirnov.test <- function(data) {
  # create empirical distribution of observed data
  ECDF <- ecdf(data)
  empiricalCDF <- ECDF(data)
  # generate test statistic
  D <- max(abs(empiricalCDF - pnorm(data)))
  # calculate p-value
  p.value <- 2 * (1 - pnorm(D * sqrt(length(data))))
  # return test statistic and p-value
  return(c(D = D, p.value = p.value))
}

set.seed(123)
dat <- rcauchy(1000, location = 0, scale = 1)

kolmogorov.smirnov.test(dat)
```

```

This will return the test statistic and the p-value. If the p-value is below a certain threshold (e.g. 0.05), then the hypothesis that the empirical distribution of the data matches the normal distribution can be rejected.
```

```
             D                    p.value
      1.347281e-01        2.039925e-05
```

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.

```
set.seed (123)
data <- data.frame(x = runif(200, 1, 10))
data$y <- 0 + 2.75*data$x + rnorm(200, 0, 1.5)
```

```
#Question 2

Heres an example of how to estimate an OLS regression in R using the Newton-Raphson algorithm with BFGS, and how to compare the results to those obtained using the lm function:

  # Set the seed
  set.seed(55)

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

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

# Estimate OLS regression using the BFGS algorithm
library(numDeriv)

# Define the negative log-likelihood function
neg_log_lik <- function(b, X, y) {
  mu <- X %*% b
  -sum(dnorm(y, mean = mu, sd = 1.5, log = TRUE))
}

# Define the gradient of the negative log-likelihood function
grad_neg_log_lik <- function(b, X, y) {
  mu <- X %*% b
  t(X) %*% (mu - y) / 1.5^2
}

# Define the starting values for the parameters
```

```r
b0 <- c(0, 0)

# Fit the model using BFGS
bfgs_fit <- optim(b0, neg_log_lik, X = cbind(1, data$x), y = data$y,
                  method = "BFGS", gr = grad_neg_log_lik)

# Compare results
cbind(lm_fit$coefficients, bfgs_fit$par)

The results obtained using the lm function and the BFGS algorithm should be equivalent.
In this case, both methods estimate the intercept and slope coefficients to be very close to 0 and 2.75, respectively.
```

```
                 [,1]      [,2]
(Intercept) 0.2041498 0.2042993
x           2.7134479 2.7134300
```