Gov 2018 Quiz: Newton-Raphson Method & Post Double Selection

Sooahn Shin

February 15, 2022

Please upload answers to all questions (including computational questions, any related graphics and R code *with comments*), in .rmd *and* knitted pdf forms to Gradescope by Thursday 11:59 pm. For any problems that require calculations, please show your work. Finally, note that only clarification questions can be asked for the quiz and you cannot consult your answer with other students.

Part 1: Newton-Raphson Method

In this part, we seek to solve the optimization problem $\min_x F(x)$ where $F : \mathbb{R} \to \mathbb{R}$ is a twice differentiable function. We will use Newton-Raphson method to find a solution for $f(x) = 0$ (i.e., the root(s) of $f(x)$) where f is the derivative of F.

Question 1: Newton-Raphson method function (2 pts)

Write a function `newton.raphson` with the following characteristics:

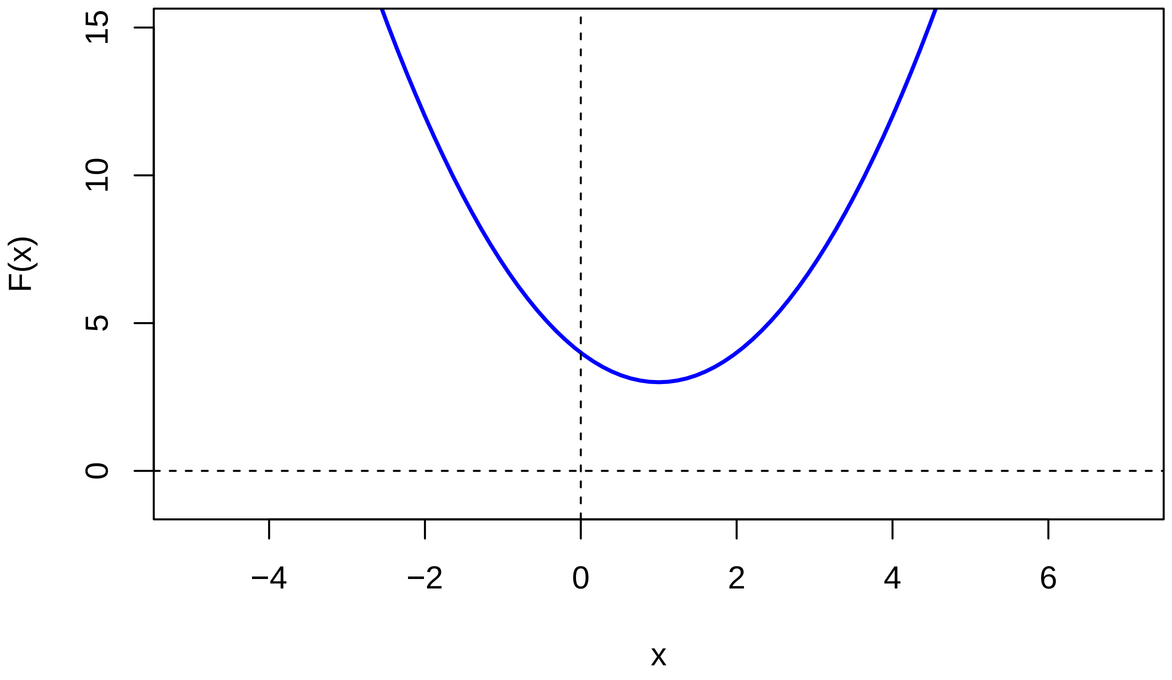
- 4 arguments:
  - f: Function for which we are searching for a solution $f(x) = 0$.
  - x_0: Initial guess for a solution $f(x) = 0$.
  - Tolerance (e.g., 1×10^{-5}): If $x_{(i+1)} - x_{(i)} <$ tolerance, then stop the iterations. In other words, stop when the improvement drops below a threshold.
  - Maximum number of iterations (e.g., 10000)
- For i-th iteration, update your guess to be $x_{(i+1)} = x_{(i)} - \frac{f(x_{(i)})}{f'(x_{(i)})}$, where f' is a derivative of f

You may use `f.prime(x)` from Problem Set 1.

```r
newton.raphson <- function(f, x, tol, max_iter) {
  n <- 0
  step <- tol + 10
  df <- function(x, d = 0.001) {
    (f(x+d) - f(x-d)) / (2*d)
  }
  while(step > tol && n < max_iter) {
    n <- n + 1
    if(df(x) == 0) {
      break
    }
    step <- -(f(x)/df(x))
    print(step)
    x <- x + step

    if(step < tol) {
      print("Meep")
    }
  }
  return(list(x, f(x), n))
}
```

Question 2: Example with one root (1 pt)

Suppose we want to find a global minimum of $F_1(x) = x^2 - 2x + 4$ (see figure below). Create a function called `f1` that computes the first derivative of F_1 at given point x. Then using the function `newton.raphson` with the initial guess of $x_0 = -2$, find the root of f_1.



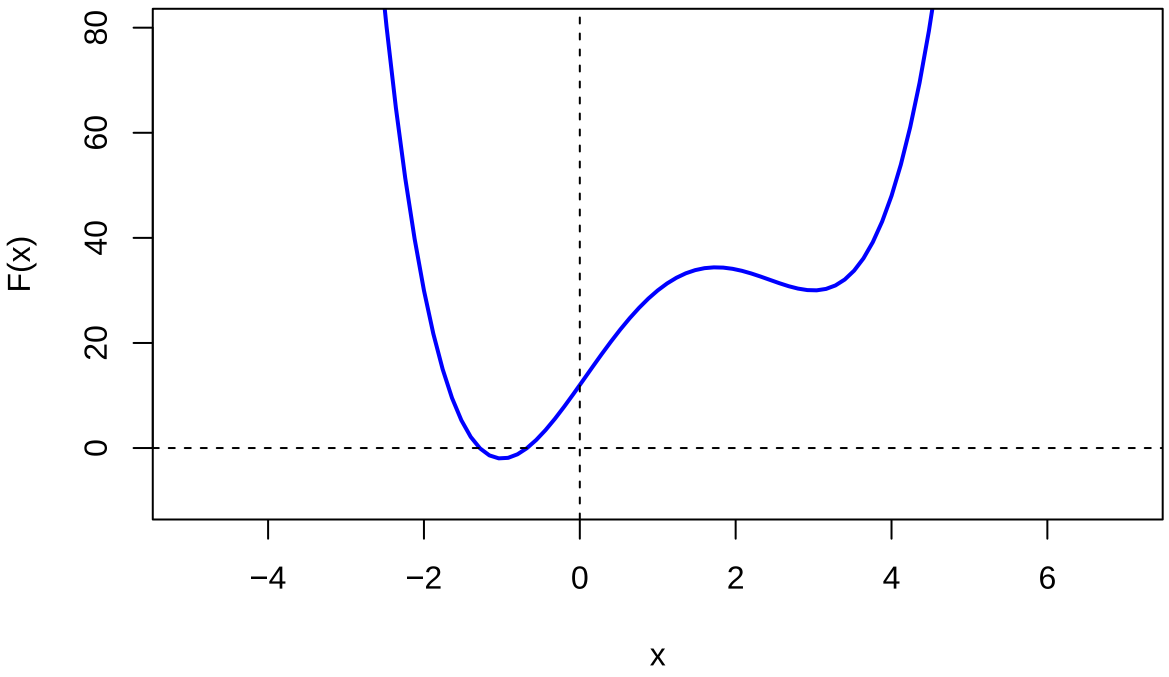
```
f1 <- function(x) {
  return((2 * x) - 2)
}

newton.raphson(f1, -2, 0.00001, 1000)
```

```
## [1] 3
## [1] -3.304024e-13
## [1] "Meep"
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 0
##
## [[3]]
## [1] 2
```

Question 3: Example with two roots (1 pt)

Now, suppose we want to find a global minimum of $F_2(x) = x^4 - 5x^3 + x^2 + 21x + 30$ (see figure below). Create a function called `f2` that computes the first derivative of F_2 at given point x. Then run the function `newton.raphson` with the initival guess of $x_0 = 4$. Do you think the approximated root of the function is the global minimum of F_2? If not, what can be done to find the global minimum?

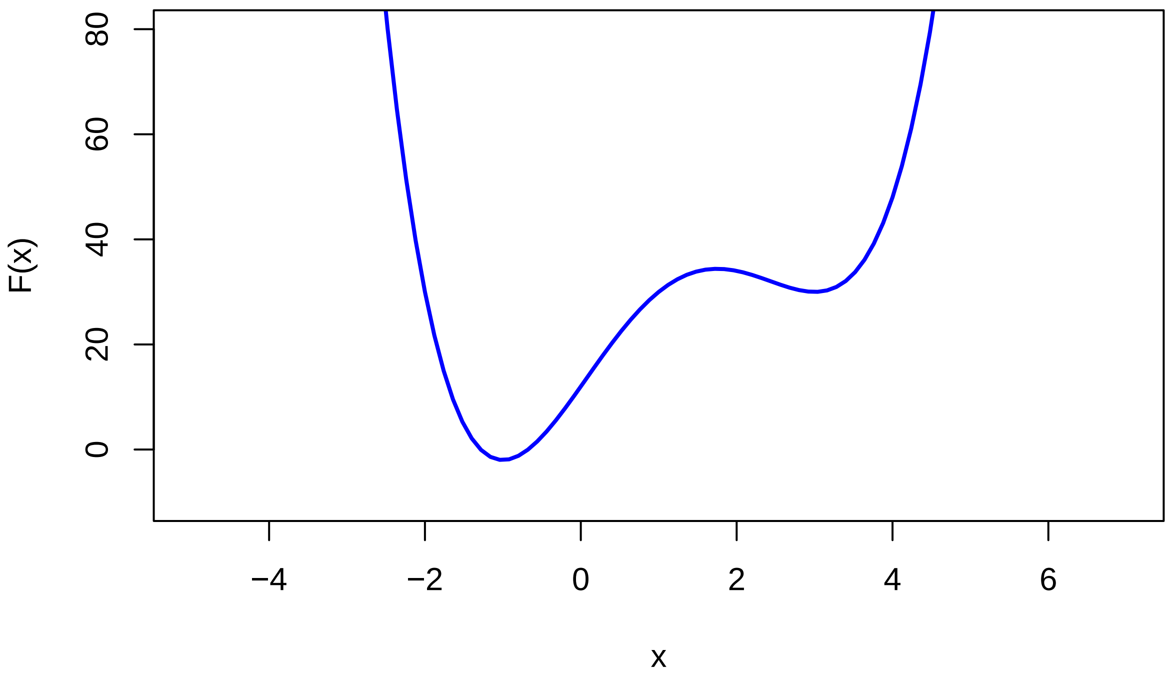


```
f2 <- function(x) {
  4*x^3 - 15*x^2 + 2*x + 21
}

newton.raphson(f2, 4, 0.00001, 1000)
```

```
## [1] -0.6081081
## [1] "Meep"
```

```
## [[1]]
## [1] 3.391892
##
## [[2]]
## [1] 11.30375
##
## [[3]]
## [1] 1
```

```
curve(F2, col = 'blue', lwd = 2, xlim=c(-5,7), ylim=c(-10,80), ylab='F(x)')
```



Part 2: Post Double Selection

In this part, we will conduct simulation study of post double selection from Belloni, Chernozhukov, and Hansen (2014). Using a synthetic data, we will investigate the following (partially) linear model with high-dimensional covariates.

$$
\begin{aligned}
Y_i &= D_i\alpha + X_i^c\beta + X_i^y\delta + \epsilon_{1i} \\
D_i &= X_i^c\gamma + X_i^e\sigma + \epsilon_{2i}
\end{aligned}
$$

where

- Y_i is the outcome variable (n by 1 vector);
- D_i is the policy (treatment) variable (n by 1 vector);
- X_i^c is a set of confounders we would like to control (X^c is n by $3p_c - 1$ matrix);
- X_i^y is a set of covariates that only has an impact on Y_i (X^y is n by $3p_y - 1$ matrix);
- X_i^e is a set of exogenous variables that only affects D_i (X^e is n by $3p_e - 1$ matrix);
- $\mathbb{E}[\epsilon_{1i} \mid D_i, X_i^c, X_i^y] = 0$ and $\mathbb{E}[\epsilon_{2i} \mid X_i^c, X_i^e] = 0$.

Under the assumption of (approximate) sparsity and some regularity conditions, we will estimate α, the impact of D_i on Y_i. We have a high-dimensional covariates $X_i = X_i^c \cup X_i^y \cup X_i^e \cup X_i^u$ from which we may select X_i^c, X_i^y, and X_i^e. Note that X_i^u does not affect D_i nor Y_i; we will use exact sparse regime instead of approximate sparse regime in the simulation study for simplicity. We will compare two different estimation strategies, one with usual post model selection and the other with post double selection. Please check the lecture slides and the paper for more details.

$$\begin{tikzcd}
X^e \arrow[dr] & & X^c \arrow[dl] \arrow[dr] & & X^y \arrow[dl] \\
& D \arrow[rr] & & Y &
\end{tikzcd}$$

A note on Causal Inference: Note that we would need additional assumptions to interpret the regression coefficient α as an unbiased estimator for an Average Treatment Effect (ATE). In this question, we will ignore the issues with identification of the causal effect and focus on investigating the post double selection method.

Question 1: Creating synthetic data (1 pt)

We will start by creating a function that generates synthetic data given true parameters. The data generating process (DGP) we will use is as follows:

1. Generate $Z^c_{i1}, \dots, Z^c_{i,p_c}$, $Z^y_{i1}, \dots, Z^y_{i,p_y}$, $Z^e_{i1}, \dots, Z^e_{i,p_y}$, and $Z^u_{i1}, \dots, Z^u_{i,p}$ as follows:
   - $Z^c_{ik} \sim \mathcal{N}(0,1)$ for $k = 1, \dots, p_c$
   - $Z^y_{ik} \sim \mathcal{N}(0,1)$ for $k = 1, \dots, p_y$
   - $Z^e_{ik} \sim \mathcal{N}(0,1)$ for $k = 1, \dots, p_e$
   - $Z^u_{ik} \sim \mathcal{N}(0,1)$ for $k = 1, \dots, p_u$
2. Let X^c_i, X^y_i, X^e_i, and X^u_i be a function of $(Z^c_{i1}, \dots, Z^c_{i,p_c})$, $(Z^y_{i1}, \dots, Z^y_{i,p_y})$, $(Z^e_{i1}, \dots, Z^e_{i,p_e})$, and $(Z^u_{i1}, \dots, Z^u_{i,p_u})$ respectively as follows:
   - $X^c_i = (Z^c_{i1}, Z^c_{i2}, \dots, Z^c_{i,p_c}, {Z^c_{i1}}^2, {Z^c_{i2}}^2, \dots, {Z^c_{i,p_c}}^2, Z^c_{i1} \cdot Z^c_{i2}, Z^c_{i2} \cdot Z^c_{i3}, \dots, Z^c_{i,p_c-1} \cdot Z^c_{i,p_c})^\top$
   - $X^y_i = (Z^y_{i1}, Z^y_{i2}, \dots, Z^y_{i,p_y}, {Z^y_{i1}}^2, {Z^y_{i2}}^2, \dots, {Z^y_{i,p_y}}^2, Z^y_{i1} \cdot Z^y_{i2}, Z^y_{i2} \cdot Z^y_{i3}, \dots, Z^y_{i,p_y-1} \cdot Z^y_{i,p_y})^\top$
   - $X^e_i = (Z^e_{i1}, Z^e_{i2}, \dots, Z^e_{i,p_e}, {Z^e_{i1}}^2, {Z^e_{i2}}^2, \dots, {Z^e_{i,p_e}}^2, Z^e_{i1} \cdot Z^e_{i2}, Z^e_{i2} \cdot Z^e_{i3}, \dots, Z^e_{i,p_e-1} \cdot Z^e_{i,p_e})^\top$
   - $X^u_i = (Z^u_{i1}, Z^u_{i2}, \dots, Z^u_{i,p_u}, {Z^u_{i1}}^2, {Z^u_{i2}}^2, \dots, {Z^u_{i,p_u}}^2, Z^u_{i1} \cdot Z^u_{i2}, Z^u_{i2} \cdot Z^u_{i3}, \dots, Z^u_{i,p_u-1} \cdot Z^u_{i,p_u})^\top$
3. Let $X_i = \begin{pmatrix} X^c_i & X^y_i & X^e_i & X^u_i \end{pmatrix}$
4. Generate regression coefficients as follows:
   - $\alpha = 0.5$
   - $\beta_j \sim \mathcal{N}(0.1, 0.01^2)$ for $j = 1, \dots, (3p_c - 1)$
   - $\delta_j \sim \mathcal{N}(0, 0.01^2)$ for $j = 1, \dots, (3p_y - 1)$
   - $\gamma_j \sim \mathcal{N}(0.8, 0.01^2)$ for $j = 1, \dots, (3p_c - 1)$
   - $\sigma_j \sim \mathcal{N}(0, 0.01^2)$ for $j = 1, \dots, (3p_e - 1)$
5. Generate random noise as follows:
   - $\epsilon_{1i} \sim \mathcal{N}(0,1)$ for $i = 1, \dots, n$
   - $\epsilon_{2i} \sim \mathcal{N}(0,1)$ for $i = 1, \dots, n$
6. Let $D_i = X^c_i \gamma + X^e_i \sigma + \epsilon_{2i}$ and $Y_i = D_i \alpha + X^c_i \beta + X^y_i \delta + \epsilon_{1i}$.

Using this DGP and the sample codes below, write a function `generate_synth_data` which generates the data and parameters.

```r
generate_synth_data <- function(n = 100, # number of observations
                                pc = 5, # dimensionality (size of dimensions) of Z_i^c
                                py = 2, # dimensionality of Z_i^y
                                pe = 2,  # dimensionality of Z_i^e
                                pu = 30,  # dimensionality of Z_i^u
                                alpha = 0.5,
                                beta = NULL,
                                delta = NULL,
                                gamma = NULL,
                                sigma = NULL,
                                seed.number = 2018
                                ) {
  set.seed(seed.number)

  #-- TODO 1: Generate Z^c, Z^y, Z^e, and Z^u (Hint: use rnorm())
  Zc = matrix(NA, nrow = n, ncol = pc)
  Zy = matrix(NA, nrow = n, ncol = py)
  Ze = matrix(NA, nrow = n, ncol = pe)
  Zu = matrix(NA, nrow = n, ncol = pu)
  # Your code here
  Zc = matrix(rnorm(n * pc), nrow = n, ncol = pc)
  Zy = matrix(rnorm(n * py), nrow = n, ncol = py)
  Ze = matrix(rnorm(n * pe), nrow = n, ncol = pe)
  Zu = matrix(rnorm(n * pu), nrow = n, ncol = pu)

  #-- TODO 2: Create X^c, X^y, X^e, and X^u
  Xc = matrix(NA, nrow = n, ncol = pc*3-1)
  Xy = matrix(NA, nrow = n, ncol = py*3-1)
  Xe = matrix(NA, nrow = n, ncol = pe*3-1)
  Xu = matrix(NA, nrow = n, ncol = pu*3-1)
  # Your codes here
  colnames(Xc) = paste0("Xc", 1:(pc*3-1))
  colnames(Xy) = paste0("Xy", 1:(py*3-1))
  colnames(Xe) = paste0("Xe", 1:(pe*3-1))
  colnames(Xu) = paste0("Xu", 1:(pu*3-1))

  #-- 3: Create X
  X = cbind(Xc, Xy, Xe, Xu)

  #-- TODO 4: Generate regression coefficients
  if (is.null(alpha)) {
    alpha = rnorm(1, mean = 0, sd = 2) # example
  }
  if (is.null(beta)) {
    # Your codes here
  }
  if (is.null(delta)) {
    # Your codes here
  }
  if (is.null(gamma)) {
    # Your codes here
  }
  if (is.null(sigma)) {
    # Your codes here
  }

  #-- TODO 5: Generate random noise
  epsilon1 = rep(NA, n)
  epsilon2 = rep(NA, n)
  # Your codes here

  #-- TODO 6: Create D and Y
  D = rep(NA, n)
```

```
  Y = rep(NA, n)
  # Your codes here
  colnames(D) = "D"
  colnames(Y) = "Y"

  # cat("## num of observations:",n,"\n## num of predictors:",ncol(X),"\n")

  res = list(X = X, D = D, Y = Y,
             alpha = alpha, beta = beta, delta = delta, gamma = gamma, sigma = sigma)
  return(res)
}
```

Question 2: Lasso methods (2 pts)

Generate synthetic data using the function `generate_synth_data` with following arguments:

```
n = 100; pc = 5; py = 2; pe = 2; pu = 30
alpha = 0.5
set.seed(2018)
beta = matrix(rnorm(pc*3-1, mean = 0.1, sd = 0.01), ncol = 1)
delta = matrix(rnorm(py*3-1, mean = 0, sd = 0.01), ncol = 1)
gamma = matrix(rnorm(pc*3-1, mean = 0.8, sd = 0.01), ncol = 1)
sigma = matrix(rnorm(pe*3-1, mean = 0, sd = 0.01), ncol = 1)
```

Estimate α using `glmnet` with two different methods: post single lasso and post double lasso. For post double lasso, use two different packages for the estimation: `glmnet` and `hdm` (the latter is written by the authors of the paper). Briefly discuss the results.

Hint: For `glmnet`, use `cv.glmnet` in place of `glmnet` — this is a cross-validated glmnet — and `coef(your-glmnet-object, s = "lambda.1se")` functions to select λ value. For `hdm`, the syntax is as follows: `hdm::rlassoEffect(x=your-X,d=your-D,y=your-Y,method="double selection")` Note that the estimates may differ between `glmnet` and `hdm` due to different default methods of selecting for λ.

Question 3: Monte-Carlo simulation (3 pts)

Note that the result from Question 2 is based on a single simulation. Now, we will conduct a Monte-Carlo simulation where we generate synthetic data multiple times (e.g., 1000 iterations) and compare the distribution of α for each method. Plot the histogram of three distributions of α (single LASSO, double LASSO with `glmnet`, and double LASSO with `hdm`). Briefly discuss the results and the difference between two methods.

Hint: Make sure to set different seed number for each iteration while using the same parameters given in the previous question.