Homework 2 - Part 2 - Math 534

Elijah Amirianfar

2024-02-14

Exercise J-2.1: A classic example of maximum likelihood estimation is due to Fisher (1925, Statistical Methods for Research Workers. Oliver and Boyd: Edinburgh.) and arises in a genetic problem. Consider a multinomial observation $\mathbf{x} = (x_1, x_2, x_3, x_4)$ with class probabilities given by

$$p_1 = \frac{2+\theta}{4}, \quad p_2 = \frac{1-\theta}{4}, \quad p_3 = \frac{1-\theta}{4}, \quad p_4 = \frac{\theta}{4}$$

where $0 < \theta < 1$. The parameter θ is to be estimated using maximum likelihood estimation based on the observed frequencies $x_1 = 1997, \quad x_2 = 907, \quad x_3 = 904, \quad x_4 = 32$

Part a) Write an R function that applies the secant method for solving this problem.

```
gradient <- function (x, teta) {
  #x=(x1,x2,x3,x4), and teta is the parameter (teta)
  dl = x[1]/(2+teta)-(x[2]+x[3])/(1-teta)+x[4]/teta}

secant_method <- function (x, teta_n_minus_1, teta_n, tolerr, tolgrad) {
  it=0
  header = c("Iteration      Theta              Mod.Rel.Error    Gradient")
  print(header, quote = FALSE)
  stop=0
  while (it<20 & stop==FALSE){
    it = it+1
    dl_n = gradient(x,teta_n)
    dl_n_minus_1 = gradient(x,teta_n_minus_1)
    # gather our gradients

    teta_n_plus_1 = teta_n - dl_n*(teta_n - teta_n_minus_1)/(dl_n-dl_n_minus_1)
    # compute our new theta value

    mod_rel_error = abs(teta_n_plus_1-teta_n)/max(1,abs(teta_n_plus_1))
    # compute modded relative error

    print(sprintf('%2.0f             %12.12f    %2.1e          %2.1e',it, teta_n_plus_1,
                  mod_rel_error, gradient(x,teta_n_plus_1)), quote = FALSE)
    if (mod_rel_error<tolerr && abs(gradient(x,teta_n_plus_1)) < tolgrad) stop=TRUE
    teta_n_minus_1 = teta_n
    teta_n = teta_n_plus_1 # Update and return our new theta's
```

```r
    }}

secant_method(c(1997,907,904,32), 0.02, 0.01, 1e-6, 1e-9)
```

```
## [1] Iteration      Theta             Mod.Rel.Error    Gradient
## [1]  1             0.024561848011    1.5e-02          4.3e+02
## [1]  2             0.027823212918    3.3e-03          2.7e+02
## [1]  3             0.033351047002    5.5e-03          6.8e+01
## [1]  4             0.035197558267    1.8e-03          1.3e+01
## [1]  5             0.035646185180    4.5e-04          7.9e-01
## [1]  6             0.035674309037    2.8e-05          9.6e-03
## [1]  7             0.035674655571    3.5e-07          6.9e-06
## [1]  8             0.035674655823    2.5e-10          6.1e-11
```

Part b) Modify the program in part a to output the following values: Iteration Number, Value of $\theta^{(n)}$, convergence ratio and number of significant digits of accuracy at iteration n.

```r
secant_method <- function (x, teta_n_minus_1, teta_n, tolerr, tolgrad) {
  it=0
  header = c("Iteration      Theta(n+1)        Convergence Ratio     Significant Digits")
  print(header, quote = FALSE)
  stop=0
  while (it<20 & stop==FALSE){
    it = it+1
    dl_n = gradient(x,teta_n)
    dl_n_minus_1 = gradient(x,teta_n_minus_1)
    # gather our gradients

    teta_n_plus_1 = teta_n - dl_n*(teta_n - teta_n_minus_1)/(dl_n-dl_n_minus_1)
    # compute our new theta value

    tstar = -1657/7680+sqrt(3728689)/7680

    convergence_ratio = abs(teta_n_plus_1-tstar)/abs(teta_n-tstar)^((1+sqrt(5))/2)
    # compute modded relative error

    mod_rel_error = abs(teta_n_plus_1-teta_n)/max(1,abs(teta_n_plus_1))
    # compute modded relative error

    significant_digits = -log(abs(teta_n_plus_1-tstar)/abs(tstar),base=10)
    # computes the amount of significant digits for each theta(n+1)

    print(sprintf('%2.0f             %12.12f    %2.3e            %2.0f',it, teta_n_plus_1,
                  convergence_ratio, significant_digits), quote = FALSE)
    if (mod_rel_error<tolerr && abs(gradient(x,teta_n_plus_1)) < tolgrad) stop=TRUE

    teta_n_minus_1 = teta_n
    teta_n = teta_n_plus_1 # Update and return our new theta's
  }}
```

```r
secant_method(c(1997,907,904,32), 0.02, 0.01, 1e-6, 1e-9)
```

```
## [1] Iteration      Theta(n+1)        Convergence Ratio     Significant Digits
## [1]  1             0.024561848011    4.162e+00             1
## [1]  2             0.027823212918    1.140e+01             1
## [1]  3             0.033351047002    5.918e+00             1
## [1]  4             0.035197558267    8.715e+00             2
## [1]  5             0.035646185180    6.738e+00             3
## [1]  6             0.035674309037    7.852e+00             5
## [1]  7             0.035674655571    7.135e+00             8
## [1]  8             0.035674655823    7.551e+00            13
```

Part c) Using part b, can you conclude that the secant method locally converges at least super linearly? How can you check that the algorithm does not converge quadratically? Explain.

```r
secant_method_modded <- function (x, teta_n_minus_1, teta_n, tolerr, tolgrad) {
  it=0
  header = c("Error Ratios")
  print(header, quote = FALSE)
  stop=0
  while (it<20 & stop==FALSE){
    it = it+1
    dl_n = gradient(x,teta_n)
    dl_n_minus_1 = gradient(x,teta_n_minus_1)
    # gather our gradients

    teta_n_plus_1 = teta_n - dl_n*(teta_n - teta_n_minus_1)/(dl_n-dl_n_minus_1)
    # compute our new theta value

    tstar = -1657/7680+sqrt(3728689)/7680

    convergence_ratio = abs(teta_n_plus_1-tstar)/abs(teta_n-tstar)^((1+sqrt(5))/2)
    # compute modded relative error

    error.ratio <- abs(teta_n_plus_1 - tstar)/abs(teta_n - tstar)
    # calculates the error ratio to determine superlinearity

    mod_rel_error = abs(teta_n_plus_1-teta_n)/max(1,abs(teta_n_plus_1))
    # compute modded relative error

    significant_digits = -log(abs(teta_n_plus_1-tstar)/abs(tstar),base=10)
    # computes the amount of significant digits for each theta(n+1)

    print(error.ratio)
    if (mod_rel_error<tolerr && abs(gradient(x,teta_n_plus_1)) < tolgrad) stop=TRUE

    teta_n_minus_1 = teta_n
    teta_n = teta_n_plus_1 # Update and return our new theta's
  }}
```

```r
secant_method_modded(c(1997,907,904,32), 0.02, 0.01, 1e-6, 1e-9)
```

```
## [1] Error Ratios
## [1] 0.4328318
## [1] 0.706522
## [1] 0.2959467
## [1] 0.2053261
## [1] 0.05967468
## [1] 0.01218047
## [1] 0.0007259626
## [1] 8.819921e-06
```

We can conclude that the secant method does converge superlinearly because the error ratios here approach 0.

Now, we will check if we have quadratic convergence. The difference is in our error ratio by taking the denominator in the error ratio in part c and squaring it.

```r
secant_method_modded_q <- function (x, teta_n_minus_1, teta_n, tolerr, tolgrad) {
  it=0
  header = c("Error Ratios")
  print(header, quote = FALSE)
  stop=0
  while (it<20 & stop==FALSE){
    it = it+1
    dl_n = gradient(x,teta_n)
    dl_n_minus_1 = gradient(x,teta_n_minus_1)
    # gather our gradients

    teta_n_plus_1 = teta_n - dl_n*(teta_n - teta_n_minus_1)/(dl_n-dl_n_minus_1)
    # compute our new theta value

    tstar = -1657/7680+sqrt(3728689)/7680

    convergence_ratio = abs(teta_n_plus_1-tstar)/abs(teta_n-tstar)^((1+sqrt(5))/2)
    # compute modded relative error

    error.ratio <- abs(teta_n_plus_1 - tstar) / abs(teta_n - tstar)^2
    # calculates the error ratio to determine quadratic convergence

    mod_rel_error = abs(teta_n_plus_1-teta_n)/max(1,abs(teta_n_plus_1))
    # compute modded relative error

    significant_digits = -log(abs(teta_n_plus_1-tstar)/abs(tstar),base=10)
    # computes the amount of significant digits for each theta(n+1)

    print(error.ratio)
    if (mod_rel_error<tolerr && abs(gradient(x,teta_n_plus_1)) < tolgrad) stop=TRUE

    teta_n_minus_1 = teta_n
    teta_n = teta_n_plus_1 # Update and return our new theta's
  }}
```

```r
secant_method_modded_q(c(1997,907,904,32), 0.02, 0.01, 1e-6, 1e-9)
```

```
## [1] Error Ratios
## [1] 16.85833
## [1] 63.57727
## [1] 37.69329
## [1] 88.36518
## [1] 125.0786
## [1] 427.8256
## [1] 2093.404
## [1] 35033.96
```

As we can see, we do not have quadratic convergence because our error ratios go to infinity. To get quadratic convergence, our error ratios need to approach a value between 0 and 1.

Part d) Finally, obtain two initial values $\theta^{(0)}$ and $\theta^{(1)}$ such that your algorithm diverges. Set maxit = 10.

```r
gradient <- function (x, teta) {
  #x=(x1,x2,x3,x4), and teta is the parameter (teta)
  dl = x[1]/(2+teta)-(x[2]+x[3])/(1-teta)+x[4]/teta}

secant_method_part_d <- function (x, teta_n_minus_1, teta_n, tolerr, tolgrad) {
  it=0
  header = c("Iteration      Theta              Mod.Rel.Error    Gradient")
  print(header, quote = FALSE)
  stop=0
  while (it<10 & stop==FALSE){
    it = it+1
    dl_n = gradient(x,teta_n)
    dl_n_minus_1 = gradient(x,teta_n_minus_1)
    # gather our gradients

    teta_n_plus_1 = teta_n - dl_n*(teta_n - teta_n_minus_1)/(dl_n-dl_n_minus_1)
    # compute our new theta value

    mod_rel_error = abs(teta_n_plus_1-teta_n)/max(1,abs(teta_n_plus_1))
    # compute modded relative error

    print(sprintf('%2.0f             %12.12f     %2.1e          %2.1e',it, teta_n_plus_1,
                  mod_rel_error, gradient(x,teta_n_plus_1)), quote = FALSE)
    if (mod_rel_error<tolerr && abs(gradient(x,teta_n_plus_1)) < tolgrad) stop=TRUE
    teta_n_minus_1 = teta_n
    teta_n = teta_n_plus_1 # Update and return our new theta's
    }}

secant_method_part_d(c(1997,907,904,32), 0.4, 0.01, 1e-6, 1e-9)
```

```
## [1] Iteration      Theta              Mod.Rel.Error    Gradient
## [1]  1             0.216253339435     2.1e-01          -1.3e+03
```

```
## [1]  2              0.144486221927    7.2e-02           -9.6e+02
## [1]  3              -0.088105919292    2.3e-01           -9.8e+02
## [1]  4              12.017083990152    1.0e+00            3.1e+02
## [1]  5              9.118403814655    3.2e-01           4.1e+02
## [1]  6              21.296778231018    5.7e-01            1.8e+02
## [1]  7              30.649948732938    3.1e-01            1.2e+02
## [1]  8              52.341102648890    4.1e-01            7.3e+01
## [1]  9              83.445591953530    3.7e-01            4.6e+01
## [1] 10              136.286354874800    3.9e-01            2.8e+01
```

Here, we let $\theta^{(0)} = 0.4$ and $\theta^{(1)} = 0.01$, and we get divergence.