Optimización inteligente

Tarea 1. Fase de Acotamiento

Ángel García Báez

2024-09-16

Breve introducción

Se planteo un ejercicio en clase donde se pedía encontrar el intervalo (espacio de búsqueda) donde sería factible encontrar el mínimo global de la función $f(x) = x^2 + \frac{54}{x}$ haciendo uso del método conocido como fase de acotamiento.

Para ello, se dio la explicación sobre que es lo que hace el método con su respectivo pseudocodigo, el reto ahora es poder programarlo en el lenguaje de programación predilecto (en este caso, en lenguaje R) para poder lograr encontrar dicho intervalo donde es más factible buscar el valor mínimo de la función en la menor cantidad de iteraciones posibles.

A continuación se muestra el pseudocodigo resumido de lo que busca hacer el algoritmo:

Método de la fase de acotamiento

Algoritmo

Paso 1: Elegir un punto inicial $x^{(0)}$ y un incremento Δ

Hacer $k = 0$

Paso 2: IF $f(x^{(0)} - |\Delta| > f(x^{(0)} + |\Delta|)$, THEN

Δ es positivo

ELSE IF $f(x^{(0)} - |\Delta| < f(x^{(0)} + |\Delta|)$

Δ es negativo

Paso 3: $x^{(k+1)} < x^{(k)} + 2^k\Delta$

Paso 4: IF $f(x^{(k+1)}) < f(x^{(k)})$ THEN

$k = k + 1$ y vuelve al paso 3.

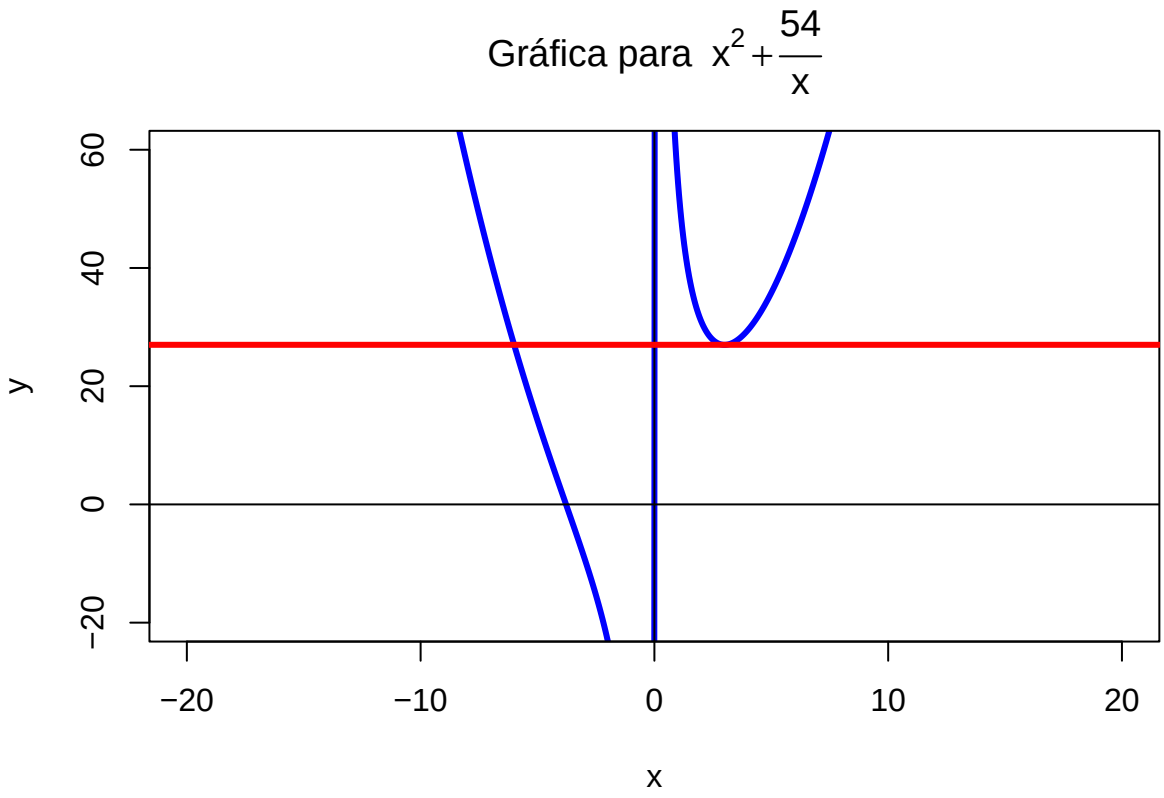
ELSE el mínimo se encuentra en el intervalo

$(x^{(k-1)}, x^{(k+1)})$
TERMINAR

Una vez explicado lo que se debe de hacer y visto la estructura del método, se realizo la implementación de dicho algoritmo (Anexo 1) en lenguaje R para ponerlo a prueba contra la función dada en un inicio.

Experimentación con los resultados.

Primero, debido a que la función se puede gráficar en un espacio de 2 dimensiones, se recurrio a esto para tener un referente visual del comportamiento de la misma:



La función presenta ciertos comportamientos interesantes, por ejemplo, a medida de que se acerca al 0 por la izquierda, dicha función tiende al infinito negativo (si buscara ahí el mínimo, nunca lo encontraría) y si se acerca al 0 por la derecha, el valor de la función tiende a ir hacia el infinito positivo (ahí tampoco es factible buscar).

Dado que la mayoría de los problemas que tienen que ver con cosas del mundo real implican que los valores sean positivos, se optara por buscar valores mayores o iguales a 0 de X tal que permitan llegar al mínimo que se ubica cuando $x = 3$ que devuelve un $f(x) = 27$.

Primer acercamiento

Durante la clase, se dieron los parámetros de inicio para la búsqueda: $x_0 = 0.5$ y $\Delta = 0.5$, los cuales al ser evaluados en el algoritmo implementado, devuelven los siguientes resultados:

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 4 iteraciones, con 10 evaluaciones de la función esta en"
```

```
## [1] 2.1 8.1
```

Bajo estos parámetros, el algoritmo encuentra un intervalo que garantiza la existencia de un mínimo, pero dicho intervalo aun es muy extenso, recordar que sabemos de antemano que el valor que minimiza la función es 3, por lo que el intervalo que devuelve es demasiado amplio para realizar dicha búsqueda, podría mejorarse.

Exploración Bajo grid search

De forma muy breve, el método de la rejilla consiste en proponer un posible conjunto de valores para los parámetros, obtener todas las posibles combinaciones de ellos e irlas evaluando una a una hasta lograr dar con cual fue la mejor combinación que cumpla con las expectativas requeridas.

Cuando son cantidades grandes de parámetros esto resulta computacionalmente muy costoso, pero para efectos prácticos, esta técnica sí puede ser implementada para este problema dado que solo son 2 parámetros de entrada y evaluar la función no resulta particularmente costoso.

Se propone el siguiente conjunto de posibles valores para X_0 y Δ:

$$X_0 = [2, 3], \qquad \Delta = [0.5, 0.1]$$

Esto da como resultado las siguientes combinaciones de parámetros que van a ser evaluadas:

$$\begin{bmatrix} x_0 & \Delta \\ 2 & 0.5 \\ 3 & 0.5 \\ 2 & 0.1 \\ 3 & 0.1 \end{bmatrix}$$

A continuación se muestran los resultados de dichas evaluaciones:

Evaluación para $x_0 = 2$ y $\Delta = 0.5$

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 3 iteraciones, con 8 evaluaciones de la función esta en"
## [1] 2.5 5.5
```

Evaluación para $x_0 = 2$" y "$\Delta = 0.1$

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 4 iteraciones, con 10 evaluaciones de la función esta en"
## [1] 2.3 3.5
```

Evaluación para $x_0 = 3$ y $\Delta = 0.5$

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 1 iteraciones, con 4 evaluaciones de la función esta en"
## [1] 2.75 3.50
```

Evaluación para $x_0 = 3$ y $\Delta = 0.1$

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 1 iteraciones, con 4 evaluaciones de la función esta en"
## [1] 2.95 3.10
```

Tras revisar los resultados encontrados por la propuesta de grid search, se observa que en cuanto se fija el valor de inicio en 3 el algoritmo tiende a converger a soluciones en menos de una iteración, a diferencia de cuando el valor inicial de $x_0 = 2$ en donde llega a tarar hasta 4 iteraciones.

Se propone otra pequeña busqueda para el valor de $x_0 = 2.9$ y $x_0 = 3$ pero ahora variando a coveniecia el valor Δ, esto con la finalidad de encontrar un intervalo más fino de busqueda.

Evaluación para $x_0 = 2.9$ y $\Delta = 0.1$

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 2 iteraciones, con 6 evaluaciones de la función esta en"
## [1] 2.9 3.2
```

Evaluación para $x_0 = 2.9$ y $\Delta = 0.01$

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 4 iteraciones, con 10 evaluaciones de la función esta en"
## [1] 2.93 3.05
```

Evaluación para $x_0 = 3$ y $\Delta = 0.01$

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 1 iteraciones, con 4 evaluaciones de la función esta en"
## [1] 2.995 3.010
```

Evaluación para $x_0 = 3$ y $\Delta = 0.0001$

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 1 iteraciones, con 4 evaluaciones de la función esta en"
## [1] 2.99995 3.00010
```

En base a este nuevo cambio de parámetros, se observa que al tomar de referencia el valor de 2.9 y hacer el Δ más pequeño, logra encontrar intervalos relativamente pequeños pero con hasta 4 iteraciones para lograrlo.

Por otro lado, al fijar el valor de x_0 a 3 y variar el Δ para valores pequeños, se observa que logra encontrar intervalos cada vez más finos donde podria estar la solución y en apenas 1 iteración.

Evaluando en valores negativos

Se propone ver que ocurre sí la función se evalua en valores negativos para observar el comportamiento de la misma:

```
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 4 iteraciones, con 10 evaluaciones de la función esta en"
## [1] -3.5  2.5
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 3 iteraciones, con 8 evaluaciones de la función esta en"
## [1] -4  2
## [1] "Delta es positivo"
## [1] "El intervalo encontrado para: 3 iteraciones, con 8 evaluaciones de la función esta en"
## [1] -4  2
```

Se observa que la función no encuentra muy buenos resultados, puesto que se atasca en valores minimos pero locales al intentar pasar por la asintota, en su rango no logra detectar la presencia del minimo que se esta buscando en el punto 3.

Conclusiones y comentarios finales

Este problema que permite observar el comportamiento de la función que se desea minimizar resulto interesante para comprender el funcionamiento del algoritmo *Fase de acotamiento*.

Para la función dada, si bien, ya se sabia de antemano que el mínimo iba a estar en 3, resulta curioso como el método que no sabe esta información a priori, llega a encontrar una región factible donde podría estar ese mínimo.

Dicho sea de paso que la definición del punto inicial es clave, puesto que si se hubiera iniciado en algún valor negativo, se corría el peligro de que el método se fuera a buscar dentro de la asintota que tiende al infinito y por ello, nunca llegara a un resultado factible, quedándose ciclado o dando una aproximación muy desatinada, o encontrando regiones para mínimos locales quizás.

Finalmente, tras las pruebas realizadas mediante un grid search relativamente pequeño, se pudo determinar que valores cercanos a 3 dan buenos resultados porque el algoritmo debe realizar apenas 1 iteración indistintamente del tamaño del Δ.

Ahora, para encontrar el intervalo más fino posible, seria necesario fijar el valor de x_0 en 3 y hacer que el valor de Δ sea lo más pequeño posible $x_0 = (0.00\ldots01)$, por lo que para efectos prácticos de esta actividad, se dejo fijado $\Delta = 0.0001$, el cual hace que el algoritmo reporte un intervalo de búsqueda en $[2.99995, 3.00010]$, encontrado tras 1 iteración y 4 evaluaciones de la función a estudiar.

Anexo 1 del codigo fuente del algoritmo

```
## rm(list=ls())
##
## #### FASE DE ACOTAMIENTO ####
##
## ### Parametros de entrada ###
## ## X_0 = punto inicial
## ## Delta = Tamaño del paso
## ## f(x) = Función
##
## # Función completa
## # Función completa
##
## FA = function(FX,x_0,Delta){
##   # Crear los contadores necesarios
##   k = 0 # Para llegar el control de las iteraciones
##   contador = 0 # Para llegar el control de las evaluaciones
##   # Evaluar esta función y guardar en un objeto para ahorrarme 1 evaluación
##   a1 = FX(x_0-abs(Delta))
##   b1 = FX(x_0+ abs(Delta))
##   contador = contador+2 # Actualizar el contador de evaluaciones de F(X)
##   # Paso 2, determinar si el incremento es positivo o negativo
##   if(a1 > b1){
##     # SI se cumple,
##     print("Delta es positivo")
##   } else {
##     if(a1 < b1){
##       # Sí se cumple, delta es negativo
##       print("Delta es negativo")
##       Delta = -Delta
##     }else{
##       # Decir que no se mueve
##       print("No encuentra cambios, vuelve al paso 1")
##       return()
##     }
##   }
##   # Paso 3 Y 4 PARA QUE BUSQUE
##   cond = T # Condición inicial que puede cambiar para detener la busqueda
##   # Defino que k sea menor o igual a 20 iteraciones
##   while (cond == T & k <=20) {
##     # Revisar si k = 0 para ahorrar una evaluación
##     # Aumentar el K y sacar el nuevo X
##     #x_0 ES el punto actual
##     x_m1 =  x_0-(2^(k-1)  * Delta)  # ES el punto k-1
##     x_k1 = x_0 + 2^k  * Delta # Es el punto k+1
##     # Evaluar el tema del paso 4
##     cond = FX(x_k1)<FX(x_0) #Re hacer la condición de pare
##     contador = contador+2 # Sumar 2 evaluaciones de la función
##     # Ahora, actualizo las variables
##     x_0 = x_k1 # Actualizo el punto actual con el punto k+1
##     # Actualizo K para seguir moviendome y buscando
##     k = k+1
##   }
```

```
##   # Reportar resultados
##   if(k <20 ){
##     # Reportar los resultados que convergieron en menos de 20 iteraciones
##     print(paste0("El intervalo encontrado para: ",k, " iteraciones, con ",
##                  contador," evaluaciones de la función esta en"))
##   }else{
##     print(paste0("El intervalo encontrado no convergio para: ",k, " iteraciones, con ",
##                  contador," evaluaciones de la función esta en:" ))
##   }
##   # Devolver el objeto intervalo que contiene al intervalo por si se necesita.
##   intervalo = c(x_m1,x_k1)
## #  print(paste0("[",intervalo[1],", ",intervalo[2],"]"))
##   return(intervalo)
##   #Termina aqui
## }
```