Optimización inteligente

Tarea 10. Método de búsqueda de patrones de Hooke-Jeeves

Ángel García Báez

2024-22-11

Contents

1 Breve introducción

Se planteo un ejercicio en clase donde se pedía encontrar el valor de x que lograra encontrar el mínimo global de la función $f(x_1, x_2) = (x_1^2 + x_2 - 11) + (x_1 + x_2^2 - 7)$ haciendo uso de el método de busqeuda de patrones de Hooke-Jeeves.

A continuación se muestra el pseudocodigo para implementar dicho método que esta dividió en 2 partes, el algoritmo que genera la exploración de patrones y el algoritmo principal que hace uso de esa exploración.

1.1 Movimiento exploratorio

Supongamos que la solución actual (el punto base) se denota por x^c. Supongamos también que la variable x_i^c es perturbada por Δ_i. Hacer $i = 1$ y $x = x^c$.

Paso 1: Encontrar $f = f(x)$, $f^+ = f(x_i + \Delta_i)$ y $f^- = f(x_i - \Delta_i)$, $f_{\min} = \min(f, f^+, f^-)$. Hacer que x corresponda a $f_{\min}$.

Paso 2: Encontrar $f_{\min} = \min(f, f^+, f^-)$. Hacer que x corresponda a $f_{\min}$.

Paso 3: ¿Es $i = N$? Si no, hacer $i = i + 1$. Ir al Paso 1}. ELSE x es el resultado. Ir al Paso 4}.

Paso 4: IF $x \neq x^c$ THEN reportar ÉXITO} ELSE reportar FRACASO}.

1.2 Método de busqueda de patrones de Hooke-Jeeves

Algoritmo

Paso 1: Elegir un punto inicial $x^{(0)}$, incrementos de las variables Δ_i $(i = 1, 2, \ldots, N)$, un factor de reducción de paso $\alpha > 1$ y un parámetro de terminación ϵ. Hacer $k = 0$.

Paso 2: Realizar un movimiento exploratorio con $x^{(k)}$ como el punto base. Hacer que x sea la salida del movimiento exploratorio. Si el movimiento exploratorio es exitoso, hacer $x^{(k+1)} = x$. GOTO Paso 4. ELSE GOTO Paso 3.

Paso 3: ¿Es $\|\Delta\| < \epsilon$? Si es CIERTO}, THEN} Terminar. ELSE hacer $\Delta_i = \Delta_i/\alpha$ para $i = 1, 2, \ldots, N$. GOTO Paso 2.

Paso 4: Hacer $k = k + 1$ y efectuar el movimiento de patrones:

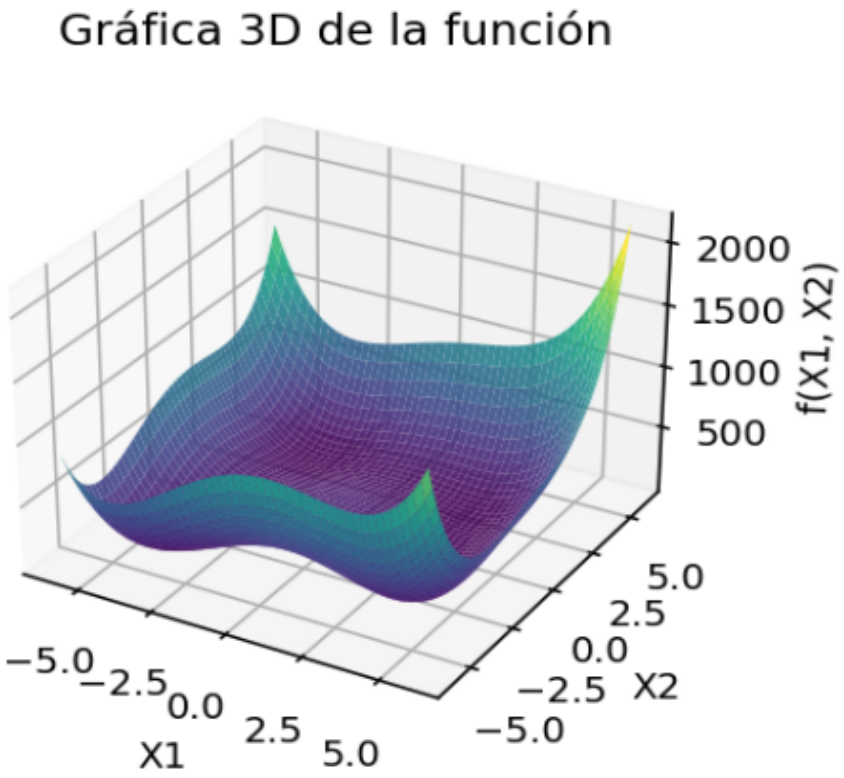
$$x_p^{(k+1)} = x^{(k)} + \left(x^{(k)} - x^{(k-1)}\right)$$

Paso 5: Realizar otro movimiento exploratorio usando $x_p^{(k+1)}$ como el punto base. Hacer que el resultado sea $x^{(k+1)}$.

Paso 6: ¿Es $f(x^{(k+1)}) < f(x^{(k)})$? Si es CIERTO}, GOTO Paso 4. ELSE GOTO Paso 3.

2 Experimentación con los resultados.

Primero, debido a que la función se puede graficar en un espacio de 3 dimensiones, se recurrió a esto para tener un referente visual del comportamiento de la misma:



La función ya no es tan fácil de visualizar o seguir, debido al aumento de una dimensión extra. Dicha función se comporta como una sabana arrugada, la cual parece tener un mínimo muy en el centro de la función con posibles mínimos locales a las orillas.

3 Experimentación

3.1 Resolución de clase

Durante la clase se abordo la forma de iniciaizar el método con los siguientes parámetros:

$$x_0 = (0, 0) \quad f(x_0) = 170 \quad \Delta = (0.5, 0.5) \quad \alpha = 2 \quad \varepsilon = 0.001$$

con el objetivo de encontrar el mínimo de la función planteada.

Una vez inicializado el algoritmo con dichos valores de x_0, α, β y ε como tolerancia, se hizo la prueba de correr el algoritmo.

A continuación se muestran los resultados de las evaluaciones que hizo el algoritmo:

X1	**X2**	**f(X)**
0.0	0.0	170.000
0.5	0.5	144.125
2.0	2.0	26.000
2.5	2.5	8.125
3.0	2.0	0.000

Para esta evaluación de clase, con los parámetros iniciales dados, ocurre que el método encuentra el valor mínimo de la función en $(3, 2)$ que devuelve un valor de 0 en apenas 5 iteraciones.

4 Conclusiones y comentarios finales

Este método presento un reto considerable puesto que la descripción de su implementación me resulto particularmente enredosa, por lo que trate de darle forma lógica a modo que llegara a la solución del mínimo aproximado encontrado en el método anterior, sin embargo sospecho que podría no estar funcionando al 100% bien pese a que se hizo un extenso trabajo de debugging pero únicamente con la función dada y los parámetros iniciales.

5 Anexo 1: Codigo principal del método de busqueda de Hooke-Jeeves.

```
## #### Función de exploración ####
## # Mecanismo generador de la exploración
## exploracion = function(fx,x0,delta){
##   # Paso 1: Encontrar f = f(x), f+ = f(xi + deltai) y f- = f(xi - deltai)
##   cont1 = 1 # Definir el contador i
##   xc = x0
##   # Calcular las dimensiones
##   D = length(xc)
##   # Acomodar los vectores y las evaluaciones
##   soluciones = matrix(0,nrow = 3,ncol = D+1)
##
##   # Paso 3. i coincide con las dimensiones? Si es el caso, ve al paso 4
##   # si no, regresa al paso 1
##   while(cont1<=D ){
##     # El primer vector es la solución inicial
##     soluciones[1:3,-(D+1)] = rbind(xc[-(D+1)],xc[-(D+1)],xc[-(D+1)])
##     # Armar la matriz de exploraciones
##     soluciones[,cont1] = rbind(soluciones[1,cont1], # Caso base
##                                soluciones[2,cont1]+delta[cont1], # CASO DONDE SE AUMENTA EL DELTA
##                                soluciones[3,cont1]-delta[cont1]) # Caso donde disminuye el delta
##     # Calcular el valor de la solución y añadir a la matriz de puntos
##     soluciones[,D+1] = apply(soluciones[,1:D],1,fx) # Aqui hubo 3 evaluaciones de la función
##     # Paso 2: Encontrar fmin = min(f,f+,f-)
##     # Reordeno en función de f(x) la matriz y me quedo el primero
##     xc = soluciones[which.min(soluciones[,D+1]),]
##     # Aumentar el contador
##     cont1 = cont1 +1
##   }
##   # Paso 4, si X encontrado es disinto de X inicial, reporta exito
##   # SI no, reporta fracaso
##   resultado = list()
##   if(sum(x0==xc[-(D+1)])==0){
##     # Si se cumple que son distintos
##     resultado[1] = "Exito"
##     resultado[2] =  list(xc)
##     return(resultado) # Reporta un exito
##   }else{
##     # Reporta un fracaso
##     resultado[1] = "Fracaso"
##   }
##   #Devolver la lista con los resultados
##   return(resultado)
## }
##
## #### Función de Busqueda Hooke Jeves principal ####
## patroneshj = function(fx,x0,delta,tol,alfa){
##   # Asignar los objetos iniciales del paso 1
##   k = 0
##   xk = x0
##   xb = x0
##   #Resultado
```

```
##   resultado = rbind(c(x0,fx(x0)))
##   # Dimensión del vector de busqueda
##   D = length(xb)
##   norma <- sqrt(sum(delta^2))
##   # Función que se itera desde el paso 2 hasta el paso 6
##   # La condición de paro esta en el paso 3, por lo que se define un
##   # while true para que busque hasta que caiga donde esta la función de paro
##   while (T){
##     # Hacer el movimiento exploratorio inicial del paso 2
##     explora = exploracion(fx,xk[1:(D)],delta)
##     # Del paso 2, revisa si se cumple la condición de siga, si no se cumple
##     # Entra en la condición del paso 3
##     if(explora[[1]]!="Exito"){
##       if(norma<tol){
##         break
##       }else{ # SI no es el caso de que falla y la norma es menor a tol
##         delta = delta/2
##         norma <- sqrt(sum(delta^2))
##       }
##     }else{
##       #Haz que la solución sea el resultado de la exploración
##       xk  = explora[[2]]
##     }
##     # Verifica la condición del paso 6
##     if(xk[D+1]<fx(xb)){
##       # Hacer que k  sea k + 1
##       k = k+1
##       # Movimiento del patron
##       xkp = 2*xk[1:(D-1)] - xb[1:D]
##       # Guarda el xk anterior en xb
##       xb = xk
##       # Actualiza el valor
##       resultado = rbind(resultado,xb)
##       # Explorar un nuevo movimiento con xkp
##       # Actualiza el xk
##       explora = exploracion(fx,xkp,delta)
##       if(explora[[1]]=="Exito"){
##         xk = explora[[2]]
##       }
##     }
##   }
##   #Muestra la matriz de resultados
##   resultado = rbind.data.frame(resultado)
##   # Cambiar los nombres para más estetica
##   names(resultado) = c(paste0("X",1:D),"f(X)")
##   return(resultado)
## } # Aqui termina la función
```

6 Anexo 2: Código para hacer las evaluaciones del método.

```
## rm(list=ls())
## # Busqueda de patrones de Hooke-Jeeves
## # Librería para hacer las tablas bonitas
## library(flextable)
## # Llamo el código
## source("Tarea 10. Hooke-Jevees MAIN.R")
## # Declaro la función que se desea minimizar
## #### Evaluación para el ejercicio de clase ####
## # x0 = (0,0)
## # delta = (0.5,0.5)
## # alfa = 2
## # tol = 0.001
## # Declaro la función que se desea minimizar
## fx = function(X){(X[1]^2+X[2]-11)^2 + (X[1] + X[2]^2-7)^2}
## #### Evaluación para el ejercicio de clase ####
## x0 = c(0,0)
## delta = c(0.5,0.5)
## alfa = 2
## tol = 0.001
## patroneshj(fx,x0,delta,alfa,tol)
```