Optimización inteligente

Tarea 9. Método de busqueda SIMPLEX Nelder y Mead

Ángel García Báez

2024-15-10

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 SIMPLEX o S^2

A continuación se muestra el pseudocodigo para implementar dicho método

1.1 Método SIMPLEX de Nelder y Mead

Algoritmo

Paso 1: Elegir $\gamma > 1, \beta \in (0, 1)$ y una tolerancia ϵ.

Paso 2: Encontrar $\mathbf{x}_h$ (el peor punto),

$\mathbf{x}_l$ (el mejor punto), y $\mathbf{x}_g$ (el segundo peor punto). Calcular: $\mathbf{x}_c = \frac{1}{N} \sum_{i=1, i \neq h}^{N+1} \mathbf{x}_i$ (centroide).

Paso 3: Calcular el punto reflejado $\mathbf{x}_r = 2\mathbf{x}_c - \mathbf{x}h$

Hacer $\mathbf{x}_{new} = \mathbf{x}_r$

IF $f(\mathbf{x}_r) < f(\mathbf{x}_l)$ THEN $\mathbf{x}_{new} = (1 + \gamma)\mathbf{x}_c - \gamma\mathbf{x}_h$ (expansión)

ELSE IF $f(\mathbf{x}_r) \geq f(\mathbf{x}_h)$ THEN $\mathbf{x}_{new} = (1 - \beta)\mathbf{x}_c + \beta\mathbf{x}_h$ (contracción)

ELSE IF $f(\mathbf{x}_g) < f(\mathbf{x}_r) < f(\mathbf{x}_h)$

THEN $\mathbf{x}_{new} = (1 + \beta)\mathbf{x}_c - \beta\mathbf{x}_h$ (contracción)

Calcular $f(\mathbf{x}_{new})$ y reemplazar $\mathbf{x}_h$ por $\mathbf{x}_{new}$

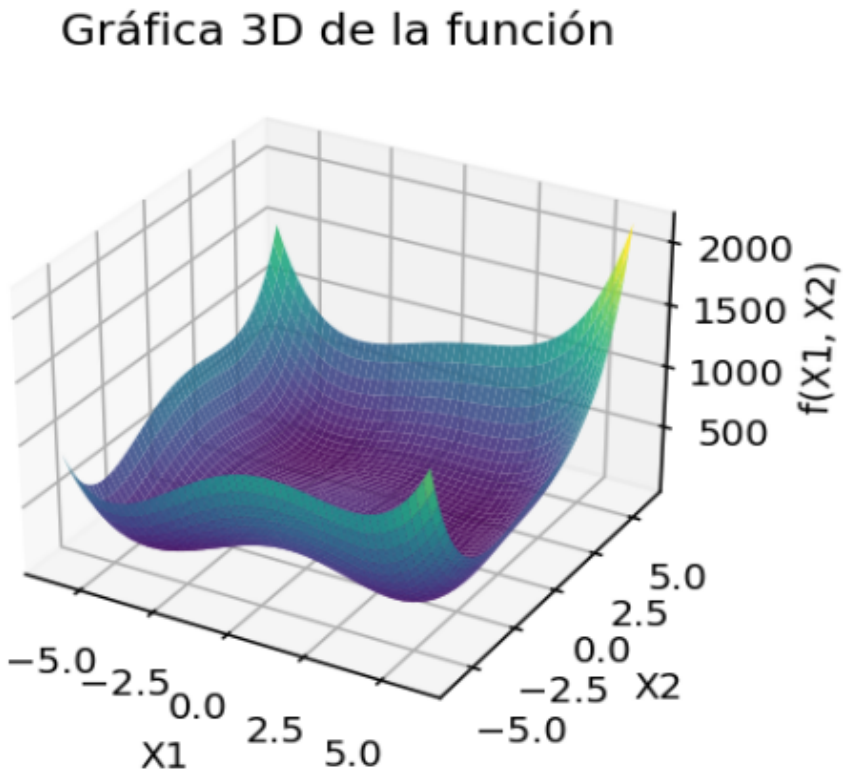
Calcular $Q = \left[\sum_{i=1}^{N+1} \frac{(f(\mathbf{x}_i) - f(\mathbf{x}_c))^2}{N+1}\right]^{1/2}$

Paso 4: IF $Q \leq \epsilon$ THEN Terminar

ELSE GOTO Paso 2.

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 facil de visualizar o seguir, debido al aumento de una dimensión extra. Dicha función se ocmporta como una sabana arrugada, la cual parece tener un minimo muy en el centro de la función con posibles minimos 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:

$$\begin{array}{ccc} x_1 = (0,0) & x_2 = (2,0) & x_3 = (1,1) \\ f(x_1) = 170 & f(x_2) = 74 & f(x_3) = 106 \\ \gamma = 1.5 & \beta = 0.5 & \varepsilon = 0.001 \end{array}$$

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

Una vez inicializado el algoritmo con dichos valores de $x_1, x_2, x_3, \gamma, \beta$ y ε como tolerancia fue que se probo el método.

A continuación se muestran unicamente los resultados del mejor punto en cada iteración, así como su valor de Q y su valor $f(x)$ tras su evaluación, todos los valores redondeados a 4 decimales.

Iter	X1	X2	Y	Q
1	2.0000	0.0000	74.0000	45.0108
2	3.7500	1.2500	21.4453	40.4022
3	3.7500	1.2500	21.4453	28.4704
4	3.7500	1.2500	21.4453	6.0937
5	2.8281	1.2344	10.1364	12.5825
6	2.8281	1.2344	10.1364	6.3244
7	3.3047	1.0391	7.7633	1.7523
8	3.3301	2.2090	6.7460	2.8964
9	3.0728	1.4292	3.5685	3.6546
10	3.0728	1.4292	3.5685	0.8483
11	3.2465	1.8191	2.0438	1.4242
12	3.0195	1.9167	0.0952	1.0093
13	3.0195	1.9167	0.0952	0.6076
14	3.0195	1.9167	0.0952	0.3592
15	3.0195	1.9167	0.0952	0.1766
16	3.0195	1.9167	0.0952	0.0964
17	2.9723	1.9697	0.0603	0.0380
18	2.9932	2.0160	0.0039	0.0293
19	2.9932	2.0160	0.0039	0.0182
20	2.9932	2.0160	0.0039	0.0080
21	2.9932	2.0160	0.0039	0.0041
22	2.9932	2.0160	0.0039	0.0021
23	3.0031	1.9948	0.0005	0.0020
24	3.0031	1.9948	0.0005	0.0004

Iter	X1	X2	Y	Q

El algoritmo logra llegar a una posible solución situada en el punto $(3.0031, 1.9948)$ que tiene un valor de $f(x, y) \approx 0.0005$ que logra satisfacer el criterio de tolerancia de 0.001 en 24 iteraciones.

4 Conclusiones y comentarios finales

El método logra encontrar una muy buena aproximación del mínimo sin perderse tanto en el espacio de busqueda, aunque depende de los puntos iniciales y de los parametros datos de β y γ.

5 Anexo 1: Código fuente del algoritmo SIMPLEX

```
## ##### Método Simplex ####
## # Introducir 3 puntos
## #x1 = un punto de p dimensiones
## #x2 = un punto de p dimensiones
## #x3 = un punto de p dimensiones
## #gamma = parametro de ajuste
## #beta = parametro de ajuste
## #tol =  tolerancia permitida
## #fx = función de varias variables
## SIMPLEX = function(fx,x1,x2,x3,gamma,beta,tol){
##   #### PASO 1: Elegir Gamma > 1 y Beta en [0,1] y una tolerancia epsilon
##   k = 1
##   #### PASO 2: ####
##   # xl el mejor punto (entiendase mejor por el que minimiza la función)
##   # xg el segundo mejor punto
##   # Encontrar Xh (el peor punto), Basados en su desempeño
##   X = rbind(x1,x2,x3)
##   Fi = apply(X,1,fx)
##   # Agregarle sus valores
##   X = cbind(X,Y = Fi)
##   Q = 50
##
##   resultados = data.frame(Iter = 0,Y = 0,Q = 0)
##   # Pegarle un punto X
##   resultados = cbind.data.frame(as.data.frame(t(x1)) ,resultados)
##   # Cambiarle los nombres
##   names(resultados)[1:(ncol(resultados)-3)]=paste0("X",1:(ncol(resultados)-3))
##   # Re acomodar para poner al principio la iter
##   it = (length(resultados)-2)
##   resultados = resultados[,c(it,setdiff(1:ncol(resultados),it ))]
##   resultados
##
##
##   while(Q>tol){
##     #### AQUI VA LA PARTE ITERATIVA
##     # Ordenarlos de acuerdo a sus desempeños
##     X = X[order(X[,ncol(X)]),]# Ordenados: 1 mejor, 2 segundo mejor, 3 peor
##     # Obtener el centroide sin el peor punto (xh)
##     xc = apply(X[-3,-ncol(X)],2,mean)
##     fxc = fx(xc)
##     #### Paso 3 ####
##     xr =  2*xc-X[3,-ncol(X)]
##
##     # Definir un x auxiliar
##     xnuevo = xr
##     # Calcular fx de xr
##     fxr = fx(xr)
##     # Verifiicar si f(xr) < f(xl) ENTONCES Xnew = (1+gamma)xc - gamma(xh) Expansión
##     if(fxr<X[1,ncol(X)]){
##       # Aplicar la expansión
##       xnuevo = (1+gamma)*xc - gamma*(X[3,-ncol(X)])
##     }else{
```

```
##       if(fxr >= X[3,ncol(X)]){
##         # Aplica el método de contracción
##         xnuevo = (1-beta)*xc + beta*(X[3,-ncol(X)])
##       }else{
##         if((X[2,ncol(X)] < fxr) & (fxr< X[3,ncol(X)])){
##           # Aplica el método de contracción
##           xnuevo = (1+beta)*xc - beta*(X[3,-ncol(X)])
##         }
##       }
##     }
##     # Calcula f(xnuevo) y remplaza xh por xnuevo
##     # Pegar como vector
##     X[3,] = c(xnuevo,fx(xnuevo))
##     # Calcular Q
##     Q = 0
##     for(i in 1:nrow(X)){
##       # Calcular las diferencias y guardarlas
##       Q = Q + ((X[i,ncol(X)]-fxc)^2/nrow(X))
##     }
##     Q = sqrt(Q)
##     # Armar la matriz
##     resultados[k,] = c(k,X[1,],Q)
##     # Actualizar K para llevar el control de las iteraciones
##     k = k+1
##   }
##   # Devolver los resultados
##   return(resultados)
## }
```

6 Anexo 2: Código para hacer las evaluaciones del método

```
##
## ### Método SIMPLEX ####
##
## # Librería para hacer las tablas bonitas
## library(flextable)
## # Llamo el código
## source("Tarea 9. SIMPLEX MAIN.R")
## # 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 ####
## # x1 = c(0,0)
## # x2 = c(2,0)
## # x3 = c(1,1)
## # tol = 0.001
## # gamma = 1.5
## # beta = 0.5
## x1 = c(0,0)
## x2 = c(2,0)
## x3 = c(1,1)
## tol = 0.001
## gamma = 1.5
## beta = 0.5
## # Evaluar la función con la corrida de clase
## res1 = SIMPLEX(fx,x1,x2,x3,gamma,beta,tol)
## res1
```