Optimización inteligente

Tarea 11. Método de direcciones conjugadas de Powel

Ángel García Báez

2024-29-11

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 busqueda de las direcciones conjugadas de Powel.

A continuación se muestra el pseudocodigo para implementar dicho método:

1.1 Método de busqueda de direcciones conjugadas de Powell

Algoritmo

Paso 1: Elegir un punto de inicio $x^{(0)}$,una tolerancia ϵ y un conjunto de N direcciones linealmente independientes; posiblemente $s^{(i)} = e^{(i)}$ para $i = 1, 2, 3, \ldots, N$.

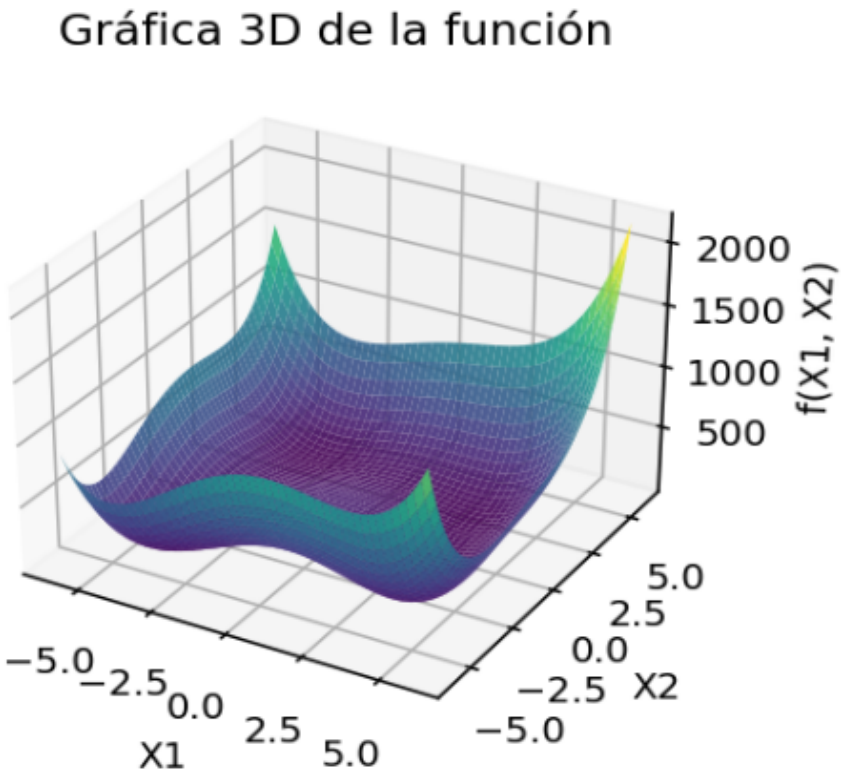
Paso 2: Minimizar a lo largo de las $N + 1$ direcciones, usando el mínimo previo para iniciar la siguiente búsqueda y haciendo que $s^{(N)}$ sea la primera y la última dirección explorada.

Paso 3: Formar una nueva dirección conjugada d usando la propiedad extendida del subespacio paralelo.

Paso 4: IF $\|d\| \leq \epsilon$ o las direcciones de búsqueda no son linealmente independientes THEN Terminar. ELSE hacer $s^{(1)} = s^{(2)}, s^{(2)} = s^{(3)}, \ldots, s^{(N-1)} = s^{(N)}$ Hacer $s^{(N)} = \frac{d}{\|d\|}$. 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 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:

$$x_0 = (0, 4) \quad f(x_0) = 130 \quad \varepsilon = 0.001$$

adicionalmente se plantearon los limites $a = -1, b = 10$ para inicializar el método de eliminación de regiones dentro del método principal

Una vez inicializado el algoritmo con dichos valores de x_0, ε, a y b se hizo la prueba de correr el algoritmo.

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

Iter	**x1**	**x2**	**fx**	**tol**
1	2.66934	3.00000	22.56773	1.15910
2	2.86140	2.07738	0.57151	0.21457
3	2.97001	1.96821	0.06883	0.04024
4	3.00117	1.97726	0.00822	0.02536
5	3.00450	1.99240	0.00105	0.00817
6	3.00203	1.99903	0.00013	0.00237
7	3.00047	2.00049	0.00002	0.00064

Para esta evaluación de clase, con los parámetros iniciales dados, el método logra encontrar un punto que aproxima bastante bien uno de los mínimos de la función. Dicho punto es $[3.00047, 2.00049]$ el cual satisface el criterio de tolerancia en apenas 7 iteraciones.

3.2 Propuesta con interalo de busqueda [-10,10]

Debido a la naturaleza tridimensional de la función, no es tan facil visualizar si es que dicha función es unimodal o multimodal. Se sospecha que la función tiene más de un minimo, por lo que se decide probar que sucede al ampliar el intervalo de busqueda del método de acotamiento para probar sí es que encuentra otro punto que se pueda considerar un minimo.

Se mantienen todos los parametros iguales a excepción del intervalo de busqueda, el cual cambia por $a = -10$ y $b = 10$

A continuación se pone a prueba el algoritmo con dichos cambios:

Iter	**x1**	**x2**	**fx**	**tol**
1	-2.80513	3.13189	0.00001	0.86893
2	-2.80513	3.13131	0.00000	0.00000

El algoritmo logra encontrar otro mínimo de la función en apenas 2 iteraciones. Dicho punto que también minimiza a la función es $[-2.80513, 3.13131]$.

4 Conclusiones

Tras la ejecución del algoritmo con los parámetros de clase y al aplicare una pequeña variación, resulta de interés el hecho de que el método aproxima bastante bien los mínimos de la función multivariable, pero es aun más llamativo el hecho de que la función puede tener más de un mínimo y que la función es capaz de encontrarlos, aunque esto lleva consigo la dificultad agregada de no poder saber con tanta facilidad si el mínimo que ya encuentra, es el mejor, se debe hacer un fino proceso se prueba con diferentes patrones para poder garantizar que el mínimo, es en efecto el mínimo que se esta buscando dadas las características que se desean satisfacer del problema.

5 Anexo 1: Codigo principal del método de busqueda de Powell

```
##
## ### Funciones de busqueda unidireccional
## source("/home/angel/Escritorio/MIAPrimerSemestre/Optimización inteligente/Tarea 2. Eliminación de reg
## source("/home/angel/Escritorio/MIAPrimerSemestre/Optimización inteligente/Tarea 4. Metodo de la secci
## # Función principal de Powell
## powell = function(fx,x,tol,a,b){
##   # Definir las funciones auxiliares internas
##   # Función que calcula la norma de un vector
##   norma = function(X){
##     # Entra un vector y se obtiene su norma
##     return(sqrt(sum(X^2)))
##   }
##   # Función para convertir el polinomio en terminos de alpha
##   fa = function(alpha) {
##     fx(x + alpha * di)  # f(x + alfa * d)
##   }
##   # Paso 1. Definir los objetos iniciales
##   # Definir un K para llevar el conteo de cuantas vueltas a dado
##   k = 0
##   # Crear una matriz de resultados
##   resultados = data.frame(Iter = 0,x1 = 0,x2 = 0,fx = 0, tol = 0 )
##   # Mecanismo generador de las direcciones linealmente independientes
##   # Crea una matriz de tamaño
##   D = length(x)
##   direcciones = matrix(0,nrow = D+1,ncol = D)
##   # Agregar la diagonal para que sean los vectores unitarios
##   diag(direcciones) = 1
##   # Valores de X para irlos almacenando
##   X = matrix(0,nrow = D+2,ncol = D)
##   # Definir que el primer punto, es el punto inicial anterior
##   X[1,] = x
##   normaD = 5 #Punto auxiliar para inicializar la norma que es condición de paro
##   while(normaD>tol){
##     # Paso 2. Minimizar a lo largo de las N + 1 direcciones, usando el minimo previo
##     # para iniciar la siguiente busqueda y haicendo que s(N) sea
##     # la primera y la ultima dirección
##     for(i in 1:nrow(direcciones)){
##       # Seleccionar la dirección i-esima
##       if(i == (D+1)){
##         di = direcciones[1,]
##       }else{
##         di = direcciones[i,]
##       }
##       # Buscar el intervalo donde se encuentra el minimo
##       elimina = ElimReg(a,b,0.001,fa)
##       ultimo = nrow(elimina)
##       # Buscar una buena aproximación del minimo
##       xmin = Dorado(fa,elimina[ultimo,2],elimina[ultimo,3],0.001 )
##       # Obtener la fila donde esta el ultimo
##       ultimo2 = nrow(xmin)
##       # Este es el alfa que minimiza la cosa
##       amin = xmin[ultimo2,2]
```

```
##       # Actualizar el punto en la matriz de puntos X
##       X[i+1,] = X[i,] + amin * di
##       x = X[i+1,]
##     }
##     # Paso 3 Forma la nueva dirección del subespacio paralelo
##     Dv = X[4,]- X[2,]
##     normaD =  norma(Dv)
##     X[1,] = x
##     ### Aqui actualiza la cosa
##     k = k + 1
##     resultados[k,] = data.frame(Iter = k,x1 = x[1],
##                                 x2 = x[2],fx = fx(x), tol = normaD)
##     # La dirección s1 ahora se convierte en la s2
##     direcciones[3,] = Dv/normaD
##     for( i in 1:(D)){
##       # Que la dirección n-1 sea igual a la dirección n
##       direcciones[i,] = direcciones[i+1,]
##     }
##   }
##   # Reportar los resultados
##   return(resultados)
## }
```

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

```
## rm(list=ls())
## # Busqueda de direcciones conjugadas de Powell
## # Librería para hacer las tablas bonitas
## library(flextable)
## # Llamo el código
## source("Tarea 11. Conjugadas de Powell MAIN.R")
## # Declaro la función que se desea minimizar
## #### Evaluación para el ejercicio de clase ####
## # fx = función multivariable
## # X = punto inicial
## # tol = tolerancia
## # a = Limite inferior de busqueda
## # b = Limite superior de busqueda
## # 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 ####
## x = c(0,4)
## tol = 0.001
## a = -1
## b = 10
##
## # Resultado de la corrida de clase forzandolo un poco
## powell(fx,x,tol,a,b)
##
## #### Evaluación con cambio de limites ####
## x = c(0,4)
## tol = 0.001
## a = -10
## b = 10
##
## # Resultado de la corrida de clase forzandolo un poco
## powell(fx,x,tol,a,b)
```