Proyecto de Optimización

Emrys Cruz Viera C-311

Facultad Matemática y Computación, Universidad de La Habana

1 Algoritmos estudiados

Los métodos quasi-Newton son técnicas utilizadas para encontrar ceros o máximos y mínimos locales de funciones, como una alternativa al método de Newton. Estos métodos pueden ser útiles si el Jacobiano o el Hessiano no están disponibles o si son demasiado costosos para calcular en cada iteración. La idea detrás de estos métodos es hacer que un algoritmo de optimización con solo un valor de función y gradiente converja más rápidamente que el descenso más pronunciado. Es decir, un método quasi-Newton no requiere un medio para evaluar la matriz Hessiana en la iteración actual, como en un método de Newton.

Los métodos quasi-Newton funcionan a través de un proceso iterativo que comienza con una suposición para el valor inicial y la inicialización de la matriz H, que es una aproximación de la Hessiana. En muchos casos esta matriz se inicializa como la matriz identidad. Luego se calcula el gradiente de la función en el punto actual y se usa la fórmula de actualización quasi-Newton para calcular el siguiente punto. Se repiten estos pasos hasta que el algoritmo converja, lo cual se suele determinar cuando el cambio en x o en el valor de la función es menor que un cierto umbral.

Los métodos quasi-Newton son métodos iterativos y pueden no converger a una solución si la suposición inicial está muy lejos del mínimo de la función, o si la función tiene múltiples mínimos locales. Por tanto, es posible que se necesite probar con diferentes suposiciones iniciales o utilizar algún método para escapar de los mínimos locales. Factor importante a tener en cuenta mas adelante.

1.1 Broyden-Fletcher-Goldfarb-Shanno (BFGS)[3]

EEn optimización numérica, el algoritmo Broyden-Fletcher-Goldfarb-Shanno (BFGS) es un método iterativo para resolver problemas de optimización no lineal sin restricciones. Al igual que el método Davidon-Fletcher-Powell, el BFGS determina la dirección de descenso precondicionando el gradiente con información de la curvatura. Lo hace mejorando gradualmente una aproximación a la matriz hessiana de la función de pérdida, obtenida sólo a partir de evaluaciones del gradiente (o evaluaciones aproximadas del gradiente) mediante un método secante generalizado. Dado que las actualizaciones de la matriz de curvatura BFGS no requieren la inversión de matrices, su complejidad computacional es sólo $\mathcal{O}(n^2)$, en comparación con $\mathcal{O}(n^3)$ en el método de Newton.

El algoritmo Broyden-Fletcher-Goldfarb-Shanno es uno de los métodos quasi-Newton más populares y su fórmula de actualización es:

$$x_{k+1} = x_k - H_k * \nabla f(x_k)$$

El método de BFGS es usado para resolver problemas optimización sobre funciones continuas y diferenciables sin restricciones, pero puede ser extendido para abordar problemas con restricciones. Cuando se trata de optimización con restricciones, una variante común del método de BFGS es el algoritmo SQP (Sequential Quadratic Programming), que combina el método de Newton para la optimización sin restricciones con técnicas de programación cuadrática para manejar restricciones. Se utiliza el método de BFGS para encontrar la dirección de búsqueda en el espacio sin restricciones, luego las restricciones se manejan mediante técnicas de programación cuadrática para encontrar una dirección de búsqueda que cumpla con las restricciones y se repite iterativamente hasta que se cumplan ciertos criterios de convergencia.

1.2 SLSQP[2]

El algoritmo SLSQP es un método iterativo utilizado para la optimización no lineal con restricciones. Este algoritmo minimiza una función de varias variables con cualquier combinación de límites, restricciones de igualdad y desigualdad. Utiliza el método cuasi-Newton de Han-Powell con una actualización BFGS de la matriz B y una función de prueba L1 en el algoritmo de longitud de paso1. Es ideal para problemas matemáticos en los que la función objetivo y las restricciones son dos veces continuamente diferenciables

2 Problemas a resolver

2.1 Problema : Dolan Function

$$f(x) = (x_1 + 1.7x_2)\sin(x_1) - 1.5x_3 - 0.1x_4\cos(x_4 + x_5 - x_1) + 0.2x_5^2 - x_2 - 1$$

$$-100 \leq x_i \leq 100$$

La solución propuesta para mínimo local en la orden del ejercicio es $f(x^*) = 0$, pero se puede notar que para $x_i = 0$ la evaluación de la función f es -1, como x_i esta en el intervalo de soluciones validas y $-1 < 0$ entonces este planteamiento no es cierto. Luego de investigar un poco llegamos a que la mejor solución conocida al problema de minimizar la función de Dolan es $x^* = (98.964258312237106, 100, 100, 99.224323672554704, 0.249987527588471)$ para la cual $f_{min}(X^*)$ =-529.8714387324576. [1]

2.1.1 Broyden-Fletcher-Goldfarb-Shanno

En un primer intento por resolver el problema aplicamos directamente el método (BFGS), el cual no fue consistente en las soluciones, en la Tabla 1 se relacionan algunos de los resultados obtenidos

Valor	x_1	x_2	X_3	x_4	x_5
−140976891.374175	$-7.99537921e+00$	3.95625495e+07	5.91487330e+07	$-1.33310217e+04$	1.64154237e+04
−350545906.54584634	$-1.41371670e+01$	9.79869826e+07	5.73161512e+07	6.84192492e+04	$-3.21491783e+00$
−157901624.11578766	$-9.27395634e+01$	$-3.63088215e+07$	8.84016567e+07	$-4.53472057e+04$	3.70191852e+01

Table 1: Resultados obtenidos con BFGS

Como podemos observar los resultados no son consistentes pero aun así son valores extremadamente pequeños. Es válido aclarar q estas pruebas se hicieron con el valor de x_0 tomado en aleatorio.

Las soluciones obtenidas se salen de las restricciones del problema lo q es señal de q dicho algoritmo no soporta restricciones. Luego decidimos optar por una variante del mismo de nombre SLSQP.

2.1.2 SLSQP

La de decisión de usar este algoritmo está dada por la intención de usar uno de la familia de los SQPs q son variantes del algoritmo BFGS que si soportan restricciones, ademas de que es la implementación q tiene Scipy.

Al probar con este algoritmo al tener la eleccion del x_0 en aleatorio las soluciones seguían sin ser concistentes pero al menos mas cercanas a lo que buscabamos.

Valor	x_1	x_2	X_3	x_4	x_5
−303.7977374613683	−80.11497731	−100	100	−26.99561043	0.25052928
−246.61924136783085	−17.28356477	−100	100	83.50940173	−0.25004254
−518.39257397499	92.67312449	100	100	100	5.03662707

Table 2: Resultados obtenidos con SLSQP

Luego decidimos probar con distintos valores de x_0 para intentar forzar una mejor solución. Primeramente probamos con $x_0 = 0$

Valor	x_1	x_2	X_3	x_4	x_5
−413.6432087084179	−7.84721045	100	100	5.15579021	−0.2452442

Table 3: Resultados obtenidos con SLSQP $x_0 = 0$

Ahora este resultado lo obtenemos de manera consistente, pero no es el deseado aun. Luego decidimos probar con $x_0 = (100, 100, 100, 100, 100)$, teniendo en cuenta que es un valor bastante cercano a donde esta el mínimo teórico.

Valor	x_1	x_2	X_3	x_4	x_5
-526.1018362465275	98.96425854	100	100	61.53137573	-0.2499748

Table 4: Resultados obtenidos con SLSQP $x_0 = (100, 100, 100, 100, 100)$

Luego este es el mejor valor al q logramos llegar y a continuación relacionamos la cantidad de iteraciones y el tiempo en segundos que tomó al algoritmo llegar a dicha solución,

Tiempo en segundos	**Iteraciones**
0.046302080154418945	14

Table 5: Evolución Diferencial

References

[1] Ali R. Al-Roomi. Unconstrained single-objective benchmark functions repository.

[2] Qiskit Community. Slsqp optimizer, 2024. Accessed: 2024-08-31.

[3] Wikipedia contributors. Broyden–fletcher–goldfarb–shanno algorithm, 2024. Accessed: 2024-08-31.