Universidad EAFIT
Escuela de ciencias aplicadas e ingeniería

Optimización II

Laboratorio I

Semestre 2023-1

Julieth Stefanny Escobar Ramírez, Sara Gallego Villada y Juan José Sánchez Sánchez

1 Pregunta 1

1. Considere la siguiente función

$$minf(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

a) Averiguar sobre el método BFGS (Broyden, Fletcher, Goldfard, Shano) y sus aplicaciones.

b) Describir el algoritmo del método BFGS.

c) Considerar como punto de partida el punto $(-1.2, 1)$ e implementar el algoritmo del gradiente descendente, el algoritmo de Newton y el algoritmo BFGS para la función f dada anteriormente. Realice una tabla donde se comparen las normas $||x_n - x_{opt}||$, donde x_n representa la sucesión de cada método y $x_{opt} = (1, 1)^T$ el valor óptimo del problema de minimización de la función f

1.1 Solución

El algoritmo BFGS es un conocido método quasi-Newton para resolver problemas de optimización sin restricciones debido a su rápida convergencia y solución numérica.

Algunas de sus aplicaciones son las siguientes:

1. Machine Learning: Es ampliamente utilizado en este campo, especialmente en el entrenamiento de redes neuronales
2. Finanzas: Se puede utilizar para optimizar las estrategias de gestión de portafolios y para modelar datos financieros
3. Física: Se puede utilizar para resolver una variedad de problemas de física, como modelar el comportamiento de los sistemas mecánicos cuánticos.
4. Ingeniería: Se puede utilizar en la optimización del diseño como el análisis estructural y el diseño aerodinámico.
5. Química: Se utiliza en simulaciones de dinámica molecular para optimizar la geometría de las moléculas.

Empecemos definiendo el algoritmo.

En primer lugar, hay que considerar el problema de optimización sin restricciones

$$minf(x), \qquad x \in R^n$$

Ahora, conocemos que la definición para los algoritmos descendientes, la cual es:

$$x_{n+1} = x_n \lambda_n d_n \ , \ n = 1, 2, 3, ...,$$

Siendo d_n la dirección de descenso y λ_n el tamaño de paso, este λ se puede definir ya sea con las condiciones Wolfe(Se definirá más adelante).
En nuestro el algoritmo obtenemos una sucesión $\{x_n\}$ con el esquema iterativo de los algortimos descendentes, en el que d_n se define solucionando una ecuación lineal:

$$B_n \cdot d + \Delta g_k$$

Entonces:

$$d_n = -B_n^{-1} \cdot g_k$$

Donde g_k es el gradiente de la función y B_n es una matriz definida positiva.

Además de esto B_n es una aproximación de la matriz Hessiana, la cual es actualizada en cada iteración del algoritmo como se muestra acontinuación:

$$B_{n+1} = B_n - \frac{B_n s_n s_n^T B_n}{s_n^T B_n s_n} + \frac{y_n y_n^T}{s_n^T y_n} \tag{1}$$

Donde

$$s_n = \lambda_n d_n \text{ y } y_n = g_{n+1} - g_k$$

La ecuación 1 se obtiene de la siguiente forma:

$$B_{n+1}(\mathbf{x}_{n+1} - \mathbf{x}_n) = \nabla f(\mathbf{x}_{n+1}) - \nabla f(\mathbf{x}_n)$$

Ahora, sea

$$\mathbf{y}_n = \nabla f(\mathbf{x}_{n+1}) - \nabla f(\mathbf{x}_n)$$

y

$$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_n$$

.

Entonces, B_{n+1} satisface:

$$B_{n+1}\mathbf{s}_n = \mathbf{y}_n$$

Que es la ecuación secante.[2]

La condición de curvatura $\mathbf{s}_k^T \mathbf{y}_n > 0$ debe cumplirse para que B_{n+1} sea definida positiva, lo cual puede verificarse pre-multiplicando la ecuación secante con $\mathbf{s}_n^T$. Si la función no es estrictamente convexa, entonces la condición debe imponerse explícitamente, por ejemplo, encontrando un punto $\mathbf{x}_{n+1}$ que satisfaga las condiciones de Wolfe, lo cual implica la condición de curvatura, utilizando una búsqueda en línea.

En vez de requerir que la matriz Hessiana completa en el punto $\mathbf{x}_{n+1}$ sea calculada como B_{n+1}, la Hessiana aproximada en la etapa n se actualiza mediante la adición de dos matrices:

$$B_{n+1} = B_n + U_n + V_n$$

Ambas U_n y V_n son matrices simétricas de rango uno, pero su suma es una matriz de actualización de rango dos. La matriz de actualización BFGS y DFP difieren de su predecesora por una matriz de rango dos. Otro método de rango uno más simple se conoce como método de rango uno simétrico, que no garantiza la definitividad positiva. Para mantener la simetría y la definitividad positiva de B_{n+1}, se puede elegir la forma de actualización como:

$$B_{n+1} = B_n + \alpha \mathbf{u}\mathbf{u}^\top + \beta \mathbf{v}\mathbf{v}^\top$$

Al imponer la condición secante,

$$B_{n+1}\mathbf{s}_n = \mathbf{y}_n$$

Al elegir $\mathbf{u} = \mathbf{y}_n$ y $\mathbf{v} = B_n\mathbf{s}_n$, podemos obtener:

$$\alpha = \frac{1}{\mathbf{y}_n^\top \mathbf{s}_n}, \quad \beta = -\frac{1}{\mathbf{s}_n^\top B_n \mathbf{s}_n}$$

Finalmente, sustituimos α y β en $B_{n+1} = B_n + \alpha \mathbf{u}\mathbf{u}^\top + \beta \mathbf{v}\mathbf{v}^\top$ y obtenemos la ecuación de actualización de B_{n+1}:

$$B_{n+1} = B_n - \frac{B_n s_n s_n^T B_n}{s_n^T B_n s_n} + \frac{y_n y_n^T}{s_n^T y_n}$$

Con lo anterior definimos el algoritmo

Paso 0 Dado $x_1 \in R^n; B_1 \in R^{n\times n}$ Definida positiva
Sea $g_1 = \nabla f(x_1)$
if $g_1 = 0$ **then**
  Parar
else
  $n = 1$
end if

Paso 1 Defina:

$$d_n = -B_n^{-1} g_k$$

Paso 2 Realice una busqueda lineal obteniendo $\lambda_n > 0$, que satisfaga las condiciones de Wolfe

$$x_{n+1} = x_n + \lambda_n d_n$$

y

$$g_{n+1} = \nabla f(x_{k+1})$$

if $g_{n+1} = 0$ **then**
  Parar

end if

Paso 3 Defina:

$$B_{n+1} = B_n - \frac{B_n s_n s_n^T B_n}{s_n^T B_n s_n} + \frac{y_n y_n^T}{s_n^T y_n} \tag{2}$$

donde

$s_n = \lambda_n d_n$,

$y_n = g_{n+1} - g_k$

Paso 4 $n := n + 1$; volver a paso 1

1.2 Condiciones de Wolfe

Anteriormente se menciono que se debe cumplir las condiciones de Wolfe. Las cuales son un conjunto de desigualdades que pretenden encontrar la longitud de paso adecuada para los métodos iterativos de búsqueda de línea. La idea es que proporcione una reducción substancial de la función objetivo f , pero a la vez, que el tiempo para encontrar tal longitud no sea muy grande. Lo ideal sería solucionar el subproblema:

$$min\phi(\alpha) = f(x_k + \alpha p_k). \qquad \alpha > 0$$

Normalmente, es costo encontrar dicha solución, por lo que se usan líneas de búsqueda inexactas que logren la máxima reducción en f posible con un costo mucho menor, y una de las estrategias es utilizar las condiciones de Wolfe que fueron consideradas en la elaboración del código adjunto. Las condiciones resumidamente exponen lo siguiente:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k \qquad c_1 \in (0, 1)$$

Sin embargo, la condición de Armijo no es suficiente para que el algoritmo funcione como se espera, ya que, la desigualdad se cumple para cualquier valor lo suficientemente pequeño. Por ello, se introduce una segunda desigualdad que evitarán pasos inaceptablemente cortos, la cual se conoce como la condición de curvatura

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k \qquad c_2 \in (c_1, 1)$$

Normalmente se elige un valor muy bajo para c_1, en el caso de c_2, se debe tomar un valor mayor a c_1 y menor a 1. Se encontró un texto relevante en la materia llamado optimización numérica de Nocedal y Wright, el cual sugiere un valor aproximado adecuado de los parámetros para los métodos de Newton y quasi-Newton como sigue: $c_1 = 10^{-4}$ y $c_2 = 0.9$. Es importante tener en cuenta que a la hora de elegir c_2 se puede afectar la convergencia global del algoritmo de optimización. Un valor muy pequeño de c_2 puede hacer que el algoritmo sea demasiado lento para converger, mientras que un valor muy grande puede hacer que el algoritmo salte sobre el mínimo global y converja a un mínimo local. Pero eso depende más que todo del problema, por ejemplo, si la función objetivo es suave y convexa, se puede elegir un valor mayor de c_2 para acelerar la búsqueda de línea, el cual es el caso de nuestra función, por lo que se eligió convenientemente un valor bastante cercano a 1 para lograr que la función convergiera rápidamente al valor óptimo.

Para la selección del valor c_1 si se tomó el valor sugerido por los autores, de manera que no sea tan restrictiva y si haya progresión en la dirección del descenso, pero tampoco valores tan bajos que cumplan con la desigualdad expuesta anteriormente pero que garantice una disminución en la función objetivo. De igual manera, el parámetro α también se obtuvo por medio de tanteo, se partió del valor 1 y se fue probando con diferentes valores α de 0 a 1 y se obtuvo la mejor solución con 0.67. La literatura guía un poco el proceso de elección, pero realmente es un proceso empírico que depende de un análisis detallado de la función objetivo y las necesidades específicas del problema de optimización que las personas expertas en el tema pueden tener en cuenta, en nuestro caso, se hizo un procedimiento más experimental para poder hallar la configuración de parámetros que permitiera resolver el problema de optimización planteado y se eligió la que dio mejores resultados

1.3 Resultados

BFGS

Recordar que se consideró $c_1 = 10^{-4}$, $c_2 = 0.92$, tamaño de paso $\alpha = 0.67$ y en 21 iteraciones se logró el sigueinte resultado:

Iteración	f(x)	norma del gradiente
19	$7.668039466937663e^{-07}$	0.038566956069111175
20	$2.4728403549799866e^{-08}$	0.006925433967398778
21	$3.398822732243045e^{-13}$	$3.937352840560102e^{-6}$

```
x_opt = [1.00000058 1.00000115]
```

NEWTON

Iteración	f(x)	norma del gradiente
5	$1.852739713292112 \times 10^{-11}$	0.05597267634300933
6	$3.432646187536325 \times 10^{-20}$	$9.624794502520175 \times 10^{-6}$
7	0.0	$1.8527623879882174 \times 10^{-11}$

```
x_opt = [1. 1.]
```

GRADIENTE

Iteración	f(x)	norma del gradiente
499	0.00011487707466460229	0.02314647517822269
500	0.00011473150195242853	0.012719099614505874
501	0.0001131587039401361	0.009771326140715946

```
x_opt = [0.98942364 0.97884516]
```

Acontinuación comparamos los 3 métodos

Se comprobó que el algoritmo que converge más rápido es el de Newton, ya que su convergencia es cuadrática, lo cual es bastante útil siempre y cuando la función a optimizar sea suficientemente suave y tenga una única solución como fue el caso del

problema dado. Sin embargo, este método puede ser computacionalmente costoso si la función tiene un número grande de variables, debido a la necesidad de calcular y almacenar la matriz Hessiana. Además, el método de Newton puede fallar si la Hessiana no es definida positiva, lo que puede ocurrir en funciones no convexas. El método BFGS tiene convergencia superlineal y tardó unas cuántas iteraciones más en aproximarse al valor óptimo, es posible que se encuentre una mejor configuración de parámetros a la elegida para obtener la solución en menos iteraciones pero no es una tarea sencilla. Su ventaja radica en que a diferencia del método de Newton, el BFGS no requiere el cálculo y almacenamiento de la Hessiana, lo que lo hace más eficiente en términos de memoria y cálculo en problemas con muchas variables. El BFGS y el Newton fueron mucho más efectivos que el método del gradiente en término del número de iteraciones. Dadas ciertas funciones particulares no convexas o con singularidades podría ser más efectiva, ya que es un algoritmo más robusto, es decir, es menos propenso a divergir o a quedarse atascado en mínimos locales.

2 Pregunta 2

$$minf(x,y) = 5x^2 + 5y^2 - xy - 11x + 11y + 11$$

a) Encuentre un punto que satisfaga las condiciones necesarias de primer orden para una solución.
b) Aplique el método de descenso visto en clase para resolver este problema, en la siguiente versión: Regla de Armijo para búsqueda de tamaño de paso. Newton para búsqueda de descenso

2.1 Solución

a.
Para encontrar un punto que satisfaga las condiciones necesarias de primer orden para una solución del problema de minimización, calculamos el gradiente y lo igualamos a cero. Para esto debemos calcular las derivadas parciales de la función e igualarlas a cero:

$$\frac{\partial f}{\partial x} = 10x - y - 11 = 0$$

$$\frac{\partial f}{\partial y} = 10y - x + 11 = 0$$

Ahora resolvemos el sistema de ecuaciones resultante: despejamos x y reemplazamos en y

$$x = \frac{11 + y}{10}$$

$$10y - \frac{11 + y}{10} + 11 = 0$$

Obtenemos que y= -1 y reemplazando en x obtenemos que x= 1.
Por lo tanto el punto que satisface las condiciones necesarias de primer orden para la función es (1,-1)
b.

Regla de Armijo:

Esta es una búsqueda inexacta y se basa en calcular una longitud de paso que de un "descenso suficiente" de f en relación al valor $f(x_k)$.

El proceso algorítmico es el siguiente:

- 1. Comenzar con t = 1.
- 2. Si $f(x_k + td_k) \leq f(x_K) + \sigma_1 t \nabla f(x_k)^T d_k$ entonces definir $t_k = t$
- 3. si no, reducir t con algún criterio hasta que se cumpla la condición.

Posibles maneras de reducir t:

- 1. hacer t = t/2;
- 2. calcular $t \in [0.1t, 0.9t]$. (puede ser el punto medio del intervalo)

Implementación:

La implementación del método de Newton para búsqueda de desecenso y regla de Armijo para búsqueda de tamaño de paso, fue hecha en python de la siguiente manera:
Se realizaron 3 funciones principales:
Una primera donde se cálcula el tamaño de paso t tomando en cuenta la condición:

$$f(x_k + td_k) \leq f(x_K) + \sigma_1 t \nabla f(x_k)^T d_k$$

Una segunda función para cálcular la dirección de descenso Newton usando la formula:

$$p = -Hess[f(x)]^{-1} \cdot \nabla f(x)$$

La tercera función principal se encarga de llamar las dos funciones anteriores para hallar la dirección (p) y el tamaño de paso (t). Para finalmente calcular el nuevo x con la formula: $x_{n+1} = x_n + t \cdot -Hess[f(x_n)] \cdot \nabla f(x_n)$ que sería en nuestra implementación sería: $x = x + t \cdot p$

El resultado de la implementación fue:

```
Puntos:  [ 0.99999997 -0.99999997]
Valor óptimo:  8.881784197001252e-15
```

Se puede observar que el resultado de la implementación fueron los puntos (0.99999997, -0.99999997) esto es aproximadamente (1,-1) que es el resultado obtenido en el punto a, donde encontramos la solución que satisfaga las condiciones necesarias de primer orden, lo que nos índica que los puntos obtenidos en la implementación son correctos.

3 Pregunta 3

Considerar el siguiente problema:

$$minf(x_1, x_2, x_3) = 9 - 8x_1 - 6x_2 - 4x_3 + 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x_1x_3$$

sujeto a:

$$x_1 + x_2 + 2x_3 \leq 3, x_j \geq 0, j = 1, 2, 3$$

Estrategía:
• Usar el método quasi–Newton para resolver problemas sin restricciones (ver problema 1)).
• Usar el algoritmo Broyden–Fletcher–Goldfarb–Shanno (BFGS) para resolver problemas no lineales. Aquí es usado para actualizar las aproximaciones a la matriz Hessiana.
•En este ejercicio, penalizaciones tipo barrera logarítmica pueden generar puntos infactibles. Cambiar el término de penalización por:

$$\bar{\theta}(g) = \theta(\bar{g})$$

dónde

$$(\bar{g}) = min\{g, -\epsilon\} \ , \ \epsilon > 0$$

3.1 Solución

Se transformó el problema original con restricciones en problemas sin restricciones mediante las funciones de penalización. Esto es, se tiene un problema de la forma:

$$\begin{aligned} & minf(x) \\ sujeto\ a\ \ & g(x) \leq 0 \end{aligned}$$

Que se condensa en una función llamada de penalización que se plantea como sigue:

$$P(x; \mu) = f(x) + \mu_n \phi(g(x^n) \tag{3}$$

donde μ es el parámetro de penalización y ϕ: $(\mathbb{R}^m)^- \to \mathbb{R}$ es la función de penalización (una función de las restricciones). El problema dado propone que se plantee la función de penalización barrera inversa, la cual se formula de la siguiente manera:

$$\phi(g(x)) = \sum_{i=1}^{m} \frac{1}{-g_j(x)}, \qquad 0 < -g(x) \leq 1 \tag{4}$$

Ahora se retoma el problema original

$$minf(x_1, x_2, x_3) = 9 - 8x_1 - 6x_2 - 4x_3 + 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x_1x_3$$

sujeto a:

$$x_1 + x_2 + 2x_3 \leq 3, x_j \geq 0, j = 1, 2, 3$$

La función objetivo más una penalización de barrera inversa es:

$$P(x;\mu) = (9-8x_1-6x_2-4x_3+2x_1^2+2x_2^2+x_3^2+2x_1x_2+2x_1x_3)-\mu(\frac{1}{x_1+x_2+2x_3-3})-\mu\frac{1}{x_1}-\mu\frac{1}{x_2}-\mu\frac{1}{x_3}$$

Cuando $min\{g,-\epsilon\} = g$

Y la función objetivo más una penalización de barrera inversa es:

$$P(x;\mu) = (9-8x_1-6x_2-4x_3+2x_1^2+2x_2^2+x_3^2+2x_1x_2+2x_1x_3)-4\mu(\frac{1}{-\epsilon})$$

Cuando $min\{g,-\epsilon\} = -\epsilon$

Se implementó la solución a dicho problema, a través de uno de los algoritmos quasi-Newton llamado: Broyden–Fletcher–Goldfarb–Shanno(BFGS). Se puede visualizar detalladamente la resolución del problema en el archivo adjunto, pero los resultados se pueden resumir en la siguiente tabla:

n	μ_n	x_1	x_2	x_3	$f(x^n)$	$P(x^n,\mu_n)$	$\mu_n\phi(g(x^n))$
1	10^{-8}	0.72498987	0.66873537	0.61247525	0.916257795	0.91625777	-1.88496843×10^{-8}
2	10^{-12}	0.81183449	0.71640744	0.68998130	0.551167922	0.551167922	$6.816837196\times10^{-12}$
3	10^{-16}	0.82272112	0.74004635	0.69849018	0.4879867957	0.4879867957	$2.0844981221\times10^{-15}$
4	10^{-20}	0.83174893	0.74210765	0.71148281	0.4567439232	0.45674392329	$3.107278522\times10^{-18}$
5	10^{-24}	0.83174893	0.74210765	0.71148281	0.4567439232	0.45674392329	$3.1072785225\times10^{-22}$
6	10^{-28}	1.23945421	1.03047346	0.36344117	0.2314816620	0.2314816620	$3.089509811\times10^{-26}$
7	10^{-32}	1.35961741	0.77601041	0.43058481	0.1125813510	0.112581351	$3.0790316e\times10^{-30}$
8	10^{-36}	1.33544548	0.777087	0.44213096	0.1118260022	0.1118260022	$3.076576773\times10^{-34}$

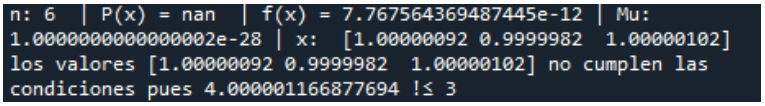
El ejercicio requería 10 iteraciones pero el algoritmo terminó antes porque cumplió la condición de parada. Se usó un principio similar al descrito en el primer punto para la selección de los parámetros en el planteamiento de las condiciones de Wolf. Dicha configuración se aproxima muy bien a la solución óptima del problema. Disminuir el épsilon podría generar que empiece a diverger, por lo cual se decidió mejor tomar aquella combinación de parámetros que nos permitiera visualizar los mejores valores de $x_1, x_2, x_3, f(x)$

Se hace la comparación con los resultados que arrojó la penalización interior que se encuentra en las notas de clase:

n	μ_n	$x_1^{(n)}$	$x_2^{(n)}$	$x_3^{(n)}$	$f(x^n)$	$P(x^n,\mu_n)$	$\mu_n\phi(g(x^n))$
1	1.00	1.164	0.732	0.314	0.354	2.415	2.061
2	10^{-1}	1.372	0.736	0.333	0.168	0.426	0.258
3	10^{-2}	1.344	0.769	0.422	0.120	0.160	0.040
4	10^{-3}	1.334	0.776	0.442	0.112	0.118	0.006

Se puede notar fácilmente que los resultados son muy similares: x_1, x_2, x_3 y $f(x)$ convergen a un resultado muy parecido al dado.

También se hizo el ejercicio de implementar la penalización de tipo barrera logarítmica, y se comprobó que efectivamente puede generar puntos infactibles. El archivo de estos resultados también se encuentra adjunto, pero la siguiente imagen ilustra claramente la situación:



Se puede observar que la solución no cumple con las restricciones del problema original, por lo cual dicha solución no es válida

4 Referencias

1. Yu-Hong Dai, Convergence Properties of the BFGS Algoritm, SIAM J. (2002)[Internet]

2. Broyden–Fletcher–Goldfarb–Shanno algorithm de Wikipedia,la enciclopedia libre[Wikipedia]