Optimización Tarea 9

Francisco Javier Peralta Ramírez

27 de abril de 2018

Resumen

En está tarea se implementó el método quasi-newton BFGS (Broyden, Fletcher, Goldfarb, y Shanno) y se probaron sus resultados para la función Rosembrock con $n \in [2, 100]$. También se probó con diferentes aproximaciones iniciales al gradiente.

1. Introducción

Los métodos Quasi-Newton son métodos que funcionan de manera muy similar al método de Newton, pero estos sólo requieren un calculo inicial de la Hessiana ya que la van actualizando con la información obtenida en cada iteración lo que elimina la necesidad de calcular la Hessiana. Para eliminar por completo la necesidad de calcular la hessiana podemos empezar el método con una aproximación a la Hessiana, la cual está dada por $A = [a_{ij}]$

$$a_{ij} = \frac{f(\boldsymbol{x} + h\boldsymbol{e_i} + h\boldsymbol{e_j}) - f(\boldsymbol{x} + h\boldsymbol{e_i}) - f(\boldsymbol{x} + h\boldsymbol{e_j}) + f(\boldsymbol{x})}{h^2}$$

donde $\boldsymbol{e_i}$ es el i-ésimo vector canónico. Podemos variar el incremento $h > 0$ para tener una aproximación inicial más cercana.

La función Rosembrock está dada por

1. Rosembrock, n = 2

$$\begin{aligned} f(\boldsymbol{x}) &= [100(x_2 - x_1^2)^2 + (1 - x_1)^2] \\ \boldsymbol{x^0} &= [-1.2, 1]^T \\ \boldsymbol{x^*} &= [1, 1]^T \\ f(\boldsymbol{x^*}) &= 0 \end{aligned}$$

Calculamos el gradiente:

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} -400(x_2 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{pmatrix}$$

2. Rosembrock, n = 100

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{i=0}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \\ \boldsymbol{x^0} &= [-1.2, 1, 1, \cdots, 1, -1.2, 1]^T \\ \boldsymbol{x^*} &= [1, 1, \cdots, 1, 1]^T \\ f(\boldsymbol{x^*}) &= 0 \end{aligned}$$

Calculamos el gradiente:

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} -400(x_2 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) - 400(x_3 - x_2^2)x_2 - 2(1 - x_2) \\ \vdots \\ 200(x_i - x_{i-1}^2) - 400(x_{i+1} - x_i^2)x_i - 2 + 2x_i \\ \vdots \\ 200(x_n - x_{n-1}^2) \end{pmatrix}$$

2. BFGS

El algoritmo *BFGS* es:

Algorithm 1 BFGS

function BFGS(x_0, F, ∇F, H, t, i)
  k := 0
  while $F(x) > t$ **and** $k < i$ **do**
    $p_k := -H_k \nabla F(x_k)$
    α_k = backtrack_alpha() ▷ Puede ser otro método
    $s_k := \alpha_k p_k$
    $x_{k+1} := x_k + s_k$
    $y_k := \nabla F(x_{k+1}) - \nabla F(x_k)$
    $\rho_k := 1/y_k^\intercal s_k$
    $H_{k+1} := (\boldsymbol{I} - \rho_k s_k y_k^\intercal) H_k (\boldsymbol{I} - \rho_k s_k y_k^\intercal) + \rho_k s_k s_k^\intercal$
    $k := k + 1$
  end while
  return x_k
end function

El algoritmo por si solo puede tener problemas numéricos cuando $y_k^\intercal s_k$ es muy pequeño o negativo, por lo que en esos casos no se actualiza H_k en la iteración.

3. Resultados

Primero se probó el algoritmo con la Hessiana obtenida de forma analítica. Para $n = 2$ el algoritmo converge muy rápido, 51 iteraciones, a un valor de $f(\boldsymbol{x}) = 9E - 11$ con $\boldsymbol{x} = (0.999990.99998)^\intercal$.

Con $n = 100$ el algoritmo converge a $f(\boldsymbol{x}) = 3.98662$ el cual es un óptimo local que ya habíamos encontrado en trabajos previos.

Para Hessianas aproximadas, usamos los valores de $h \in [0.5, 0.01, 0.0001]$

n	h	k	$\boldsymbol{x}_k$	$f(\boldsymbol{x}_k)$	cond($\boldsymbol{H}_k$)
2	0.5	3	$(-1.3311.755)$	5.457	-45.925
2	0.01	49	$(1, 1)$	2E-8	2356.85
2	0.0001	51	$(1, 1)$	1E-9	2321.38
100	0.5	3	$(0.161.351.15, \cdots)$	232.854	3.031
100	0.01	18	$(1, 1, 1, \cdots)$	1E-5	22.924
100	0.0001	32	$(-0.99, 1, 1, \cdots)$	3.986	22.671

4. Conclución

El algoritmo *BFGS* tardó más en converger que el algoritmo de *Newton*, esto es de esperarse ya que se tiene menos información, pero por otra parte *BFGS* tiene el beneficio de no tener que calcular la Hessiana en cada iteración y cuando usamos un aproximado a la Hessiana ni siquiera tenemos que tener la formula analítica de la Hessiana. El siguiente paso a tomar sería no depender del gradiente, esto haría que el algoritmo tardara más, pero podríamos equaciones donde calcular el gradiente de forma analítica sea muy dificil.

Los resultados de *BFGS* son los que esperabamos, es mucho más rápido que los algotimos de máximo descenso de gradiente y ligeramente más lento que *Newton*. Al probar con aproximaciones al Hessiano con tamaños de paso diferentes pudimos ver la importancia del tamaño de paso y que despues de cierto punto no vale la pena hacerlo más pequeño. Para nuestra sorpresa el error que introduce un tamaño de paso $h = 0.01$ en el calculo de Hessiano para Rosembrock con $n = 100$ ayudó al algoritmo a converger a la solución óptima, cosa que no se había logrado previamente ni con *Newton*.