Tarea 4. Optimización

Oscar Esaú Peralta Rosales
Maestría en Computación
Centro de Investigación en Matemáticas

***Resumen*—En esta tarea se realiza la implementación tres métodos de optimización**

- **Cubic interpolation**
- **Barzilai-Borwein**
- **Zhang-Hager**

y se evalua su comportamiento y resultados con dos tipos de funciones a optimizar; Rosembrock function, con n = 100 y Wood function.

También se utiliza el método de Barzilai-Borwein para optimizar los parámetros de para la función de máxima verosimilitud de regresión logística para dos clases y se evalua sobre el conjunto de datos de imágenes de números MNIST.

***Index Terms*—Cubic interpolation, Barzilai-Borwein, Zhang-Hager**

I. Introduction

El metodo de Cubic interpolation se basa en la suposición de que tenemos un intervalo $[a, b]$ en dónde podemos obtener el valor de algún tamaño de paso y dos tamaños de pasos anteriores α_{i-1} y α_{i-2} y una función cúbica que interpola a f en $\phi(\alpha_{i-1})$, $\phi'(\alpha_{i-1})$, $\phi(\alpha_i)$ y $\phi'(\alpha_i)$. Una vez obtenidos estos valores de α mediante interpolación elegimos como tamaño de paso (en orden) aquel que cumpla con las condiciones de Armijo, iterando y reduciendo el intervalo hasta encontrar alguno.

El método de Barzilai-Borwein se basa en encontrar un tamaño de paso α_k tal que $\alpha_k g_k$ aproxime a $H^{-1}g_k$ sin tener que calcular el Hessiano de la función. El método de Zhang-Hager combina el método anterior con una técnica de búsqueda en linea no monotónica.

Esta ocación usando el método de Barzilai-Borwein se realizó una optimización sobre los parámetros β y β_0 de la función de máxima verosimilud de regresion logística para obtener un modelo de clasificación binaria, cuya función a optimizar para un conjunto de observaciones $S = \{(x_i, y_i\}$ con $x_i \in R^{784}$ and $y_i \in \{0, 1\}$ está dada por

$$h(\beta, \beta_0) = \sum_{i=1}^{n} y_i log \pi_i + (1 - y_i) log(1 - \pi_i) \tag{1}$$

$$pi_i := pi_i(\beta, \beta_0) = \frac{1}{1 + \exp(-x_i^T \beta - \beta_0)} \tag{2}$$

Aquí cada x_i representa un vector con los valores de los pixeles de una imagen del dataset de $MINIST$ mencionado anteriormente y y_i representa la etiqueta asociada a dicha imagen.

II. Métodología

II-A. Implementación de los métodos de cubic interpolation, Barzilai-Borwein y Zhang-Hager

Para la implementación del método usando interpolación cúbica usamos la condición de Armijo $f(x_k + \alpha_k d_k) < f(x_k) + c_1 \alpha_k g_k^T d_k$ para verificar que alguna $\alpha_0, \alpha_1, \alpha_2$ la cumplan, y en donde definimos: $\phi(\alpha_k) = f(x_k - \alpha_k g_k)$, $\phi(0) = f(x_k)$, $\phi'(0) = -g_k^T g_k$ $\alpha_1 = \frac{-\alpha_0^2 \phi'(0)}{2(\phi(\alpha) - \phi'(0)\alpha_0 - \phi(0))}$, $\alpha_2 = \frac{-b + \sqrt{b^2 - 3ac}}{3a}$ con $c = \phi(0)$ y

$$\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\alpha_1^2 \alpha_0^2 (\alpha_1 - \alpha_0)} \begin{pmatrix} \alpha_0^2 & -\alpha_1^2 \\ \alpha_0^3 & \alpha_1^3 \end{pmatrix} \begin{pmatrix} \phi(\alpha_1) - \phi'(0)\alpha_1 - \phi(0) \\ \phi(\alpha_0) - \phi'(0)\alpha_0 - \phi(0) \end{pmatrix}$$

En el metodo de Barzilai-Borwein la actualización del tamaño de paso se realiza en cada iteración de la forma $s_{k-1} = x_k - x_{k-1}$, $y_{k+1} = g_k - g_{k-1}$, $\alpha_k = \frac{s_{k-1}^T y_{k+1}}{y_{k+1}^T y_{k+1}}$.

Para la implementación del algoritmo de Zhang-Hager se optó por retomar el algoritmo que usa interpolación cúbica y agregarle la modificación a la condición de Armijo mediante el parámetro c_k, por tanto la condición de Armijo ahora es $f(x_k + \alpha_k d_k) < c_k + c_1 \alpha_k g_k^T d_k$ donde en cada paso se actualiza con $q_{k-1} = q_k$, $q_k = \eta q_k + 1$, $c_k = (\eta q_{k-1} c_k + f(x_k))/q_k$.

Puesto que se requiere para algunos cálculos el α anterior para los tres algoritmos se propuso un α inicial en el rango de $1e^{-3}$.

Para la función de Rosembrock con n=100 se usó un x inicial igual a $x = [-1{,}2, 1, 1, ..., 1, -1{,}2, 1]^T$ cuyo valor óptimo es $x = [1, 1, ..., 1, 1]^T$.

Para la función de Wood se usó un x inicial igual a $x = [-3, -1, -3, -1]^T$ cuyo valor óptimo es $x = [1, 1, 1, 1]^T$.

En la siguiente sección se muestran los resultados obtenidos.

También se muestran los algoritmos implementados en el Apéndice V-B.

II-B. Optimización sobre modelo de regresión logística binario

Dado que la el modelo representado por (1) y (2) es para clasificación binaria se seleccionó solo los imágenes correspondientes a las etiquetas 0 y 1 y con ellas se procedió a optimizar los valores de β y β_0

Para una mejor implementación se procedió a reescribir las ecuaciones (1) y (2) de la siguiente forma.

Sea $\phi = [\beta, \beta_0]$ y $W = \{w_i\}$ donde $w_i = [x_i, 1]$, con ello podemos reescribir (1) y (2) como:

$$h(\phi) = \sum_{i=1}^{n} y_i log \pi_i + (1 - y_i) log(1 - \pi_i) \tag{3}$$

$$\pi_i := \pi_i(\phi) = \frac{1}{1 + \exp(-w_i^T \phi)} \tag{4}$$

Así el gradiente de h esta dado por:

$$\nabla_\phi \pi_i(\phi) = \pi_i(\phi)(1 - \pi_i(\phi)) w_i \tag{5}$$

$$\nabla_\phi h(\phi) = \sum_{i=1}^{n} y_i \frac{\nabla_\phi \pi_i(\phi)}{\pi_i(\phi)} + (1 - y_i) \frac{\nabla_\phi (1 - \pi_i(\phi))}{1 - \pi_i(\phi)}$$

$$\nabla_\phi h(\phi) = \sum_{i=1}^{n} y_i (1 - \pi_i(\phi)) - (1 - y_i) \pi_i(\phi) \tag{6}$$

Para la optimización de sobre los parámetros β y β_0 se decició usar el algoritmo de Barzilai-Borwein.

El conjunto de datos está dividido en 3 partes, el conjunto de entrenamiento, el conjunto de validación y el conjunto de prueba.

La optimización de la función se realizó usando el conjunto de entrenamiento, y para su verificación se procedió a calcular el error obtenido al clasificar el conjunto de validación mediante la siguiente fórmula:

$$error = \frac{1}{|\tau|} \sum_{x_i, y_i \in \tau} |1_{\pi_i(\phi)}(w_i) - y_i|$$

En la siguiente sección se muestran los resultados obtenidos.

III. Resultados

De las diversas ejecuciones, en las tablas I y II se muestran los mejores resultados obtenidos, para la aproximación del optimo para la función de Rosembrock y Wood respectivamente.

Cuadro I
Tabla comparativa entre los 3 métodos con la función de Rosembrock

Algoritmo	***Iteraciones***	$\|\|g_k\|\|$	$\|f(x_k) - f(x^*)\|$
Cubic Interpolation	10000	9.9977	3.9866
Barzilai-Borwein	953	10.0000	1.5070e-10
Zhang-Hager	10000	9.9977	3.9866

Usando el método de Barzilai-Borwein para la clasificación de las imágenes se obtuvo un error de 0.00047, es decir clasificó correctamente las imagenes con probabilidad de mas 0.9995 de acertar.

La figura 1 se muestra la imagen formada por la optimización de las ϕ la cual representa algo paracido a un enmascaramiento de las carácterísticas distintivas de las imagenes conformadas de ceros y unos y que ayuda a clasificarlas correctamente.

Cuadro II
Tabla comparativa entre los 3 métodos con la función de Wood

Algoritmo	***Iteraciones***	$\|\|g_k\|\|$	$\|f(x_k) - f(x^*)\|$
Cubic Interpolation	10000	1.9180	7.8769
Barzilai-Borwein	207	2.0000	3.2988e-15
Zhang-Hager	10000	1.9180	7.8769

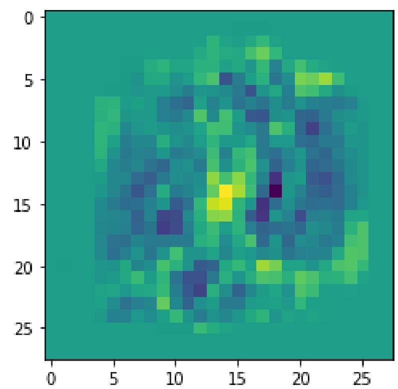

Figura 1. Máscara generada por los valores optimizados de β usada para clasificar las imágenes de números de 0's y 1's.

IV. Conclusiones

Como se ve en los resultados el método de Barzilai-Borwein fue con el que mejores aproximaciones se obtuvieron, la elección de un $\alpha = 1e - 2$ fue crucial para obtener un biuen resultado con la función de Rosembrock puesto al parecer este tamaño de paso inicial hace que salga de un mínimo local.

La clasificación de las imágenes usando el método de Barzilai-Borwein fue bastante bueno puesto que se clasificó un conjunto de datos completamente nuevo con un error aproximado de 0.00047. La imagen mostrada de la representación grafica de los β optimizado parece capturar información tal que le ayuda a decirdir a que clase pertenece, intuitivamente parace que el centro lo usa para identicar si es o no es un cero, por la forma que esta presenta.

V. Apéndice

V-A. Problemas

1: Sea $f : \mathcal{R} \to \mathcal{R}$ dado por $f(x) = (x - a)^4$, donde a es una constante. Supon que aplicamos el Método de Newton al problema de minimizar f.

- a) Desarrolla la ecuación de actualización para el Método de Newton aplicado al problema.

- b) Sea $y_k = |x_k - a|$ donde x_k es la k-ésima iteración del Método de Newton. Muestra que la secuencia $\{y_k\}$ satisface que $y_{k+1} = \frac{2}{3}y_k$.
- c) Muestra que $x_k \to a$ para cualquier valor inicial x_0.
- d) ¿Cuál es el orden de convergencia de la secuencia $\{x_k\}$?.
- e) ¿El orden de convergencia anterior contradice el teorema a cerca de la convergencia del del Método de Newton?

Solución

a: Sabemos que la dirección de descenso para una serie $\{x_k\}$ generada por el Método de Newton está dado por $d_k = -H(x_k)^{-1}g(x_k) = x_{k+1} - x_k$ por lo tanto $x_{k+1} = x_k - -H(x_k)^{-1}g(x_k)$. Para este caso $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$, donde $f'(x) = 4(x-a)^3$ y $f''(x) = 12(x-a)^2$.

Sustituyendo en la fórmula de actualización tenemos que $x_{k+1} = x_k - \frac{4(x_k-a)^3}{12(x_k-a)^2}$ y finalmente $x_{k+1} = x_k - \frac{1}{3}(x_k - a)$

b: Del punto anterior tenemos que $x_{k+1} = x_k - \frac{1}{3}(x_k - a)$, como $y_k = |x_k - a|$ entonces $y_{k+1} = |x_{k+1} - a|$, sustituyendo x_{k+1} tenemos que $y_{k+1} = |x_k - \frac{1}{3}(x_k - a) - a| = |x_k - \frac{1}{3}x_k + \frac{1}{3}a - a|$ y finalmente $y_{k+1} = \frac{2}{3}|x_k - a| = \frac{2}{3}y_k$

c: Queremos mostrar que $\lim_{k\to\infty} x_k = a$ valor inicial x_0, reescribiendo $\lim_{k\to\infty} x_k - a = 0$ o también podemos verlo como $\lim_{k\to\infty} y_k = 0$ si tomamos la siguiente iteración también debe cumplirse que $\lim_{k\to\infty} y_{k+1} = 0$ o que $\lim_{k\to\infty} \frac{2}{3}y_k = 0$.

Observemos que $y_1 = \frac{2}{3}y_0$, $y_2 = (\frac{2}{3})^2 y_0$, ... $y_k = (\frac{2}{3})^k y_0$. Luego $\lim_{k\to\infty}(\frac{2}{3})^k y_0 = 0$ puesto que $\frac{2}{3})^k$ tiene a ser cada vez más pequeño. Así $x_k \to a$ para cualquier valor inicial x_0.

d: Por definición, una sucesión $\{x_k\}$ converge a L con orden q si $\lim_{k\to\infty} \frac{|x_{k+1}-L|}{|x_k-L|^q} = \mu$ para $\mu > 0$. Ya demostramos que $x_k \to a$ y tomemos $q = 1$ por primero demostrar que tiene convergencia lineal, entonces, $\lim_{k\to\infty} \frac{|x_{k+1}-a|}{|x_k-a|} = \mu$, sustituyendo tenemos que

$$\lim_{k\to\infty} \frac{|x_{k+1}-a|}{|x_k-a|} = \lim_{k\to\infty} \frac{y_{k+1}}{y_k}$$

$$\lim_{k\to\infty} \frac{|x_{k+1}-a|}{|x_k-a|} = \lim_{k\to\infty} \frac{2y_k}{3y_k}$$

$$\lim_{k\to\infty} \frac{|x_{k+1}-a|}{|x_k-a|} = \frac{2}{3}$$

así tenemos que $\mu = \frac{2}{3} > 0$ y por tanto la serie x_k tiene orden de convergencia lineal.

e: La definición de convergencia al menos cuadrática del Método Newton es asegurada para cualquier punto suficientemente cerca a la solución y que $H(x*)$ sea invertible sin embargo notemos que la convergencia a a ha sido comprobada para cualquier punto unicial y $f''(a) = 12(a-a)^2 = 0$ por tanto no se cumple con los criterios mencionados anteriormente y no contradice la definición.

V-B. Algoritmos

Algorithm 1: Linear Search with Cubic Interpolation

```
Result: x_k
x_k <- Inicializar
g_k <- ∇f(x)
tol <- Inicializar Tolerancia
α_k <- Inicializar alpha inicial
α_0 <- α_k
α_1, α_2
while ||g_k|| > tol do
    α_0 = α_k
    if !armijo(x_k, g_k, alpha_0) then
        α_1 = computeAlpha1(x_k, g_k, α_0)
        if armijo(x_k, g_k, α_1) then
            α_k = α_1
        end
        else
            α_2 = computeAlpha2(x_k, g_k, α_0, α_1)
            while !armijo(x_k, g_k, α_2) do
                α_0 = α_1
                α_1 = α_2
                α_2 = computeAlpha2(x_k, g_k, α_0, α_1)
            end
            α_k = α_2
        end
    end
    x_k = x_k - α_k g_k
    g_k = g(x_k)
end
```

Referencias

[1] Jorge Nocedal, Stephen J. Wright, “Numerical Optimization,” Second Edition, Springer.

Algorithm 2: Barzilai-Borwein

Result: x_k
x_k <- Inicializar
x_{k-1} <- NULL
g_k <- $\nabla f(x)$
g_{k-1} <- NULL
tol <- Inicializar Tolerancia
α_k <- Inicializar alpha inicial
$k = 0$ **while** $||g_k|| > tol$ **do**
    if $k! = 0$ **then**
        $s_{k-1} = x_k - x_{k-1}$
        $y_{k+1} = g_k - g_{k-1}$
        $\alpha_k = \frac{s_{k-1}^T y_{k+1}}{y_{k+1}^T y_{k+1}}$
    end
    $x_{k-1} = x_k$
    $x_k = x_k$ - $\alpha_k g_k$
    $g_{k-1} = g_k$
    $g_k = g(x_k)$
end

Algorithm 3: Zhang-Hager

Result: x_k
x_k <- Inicializar
g_k <- $\nabla f(x)$
c_k <- $f(x)$
q_k <- Inicializar
q_{k-1} <- NULL
η <- $f(x)$
tol <- Inicializar Tolerancia
α_k <- Inicializar alpha inicial
α_0 <- α_k
α_1, α_2
while $||g_k|| > tol$ **do**
    $\alpha_0 = \alpha_k$
    if *!armijo*(x_k, g_k, $alpha_0$, c_k) **then**
        α_1 = computeAlpha1(x_k, g_k, α_0)
        if *armijo*(x_k, g_k, α_1, c_k) **then**
            $\alpha_k = \alpha_1$
        end
        else
            α_2 = computeAlpha2(x_k, g_k, α_0, α_1)
            while *!armijo*(x_k, g_k, α_2, c_k) **do**
                $\alpha_0 = \alpha_1$
                $\alpha_1 = \alpha_2$
                α_2 = computeAlpha2(x_k, g_k, α_0, α_1)
            end
            $\alpha_k = \alpha_2$
        end
    end
    $x_k = x_k$ - $\alpha_k g_k$
    $g_k = g(x_k)$
    $q_{k-1} = q_k$
    $q_k = \eta q_k + 1$
    $c_k = (\eta q_{k-1} c_k + f(x_k))/q_k$
end