Tarea 12: Método BFGS y Descenso Gradiente

Optimización I

Maestría en Computación
Centro de Investigación en Matemáticas

Esteban Reyes Saldaña
esteban.reyes@cimat.mx

14 de mayo de 2021

Resumen

En esta tarea se utilizó el método de descenso gradiente con paso exacto y el método BFGS para minimizar una función de clasificación binaria sobre el conjunto de datos *mnist.pkl.gz.* En este caso el conjunto de datos corresponde a imágenes de números dígitos escritos a mano. Se presenta a continuación una descripción general de dicha función así como el pseudocódigo de los métodos implementados. Debido a la complejidad analítica de la función, se utilizó una aproximación del gradiente y del Hessiano. En los resultados se incluyen pruebas de descenso gradiente y del método BFGS .Finalmente se incluyen conclusiones observadas a partir de la experimentación.

1 Introducción

1.1 Descenso Gradiente

Algunos resultados relevantes sobre el gradiente, vistos en la tarea pasada son

Teorema 1.1. Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable. Entonces para toda $x \in dom(f)$, $\nabla f(x)$ es perpendicular al conjunto de nivel

$$S = \{x \in \mathbb{R}^n | f(x) = c, c \text{ constante.}\}$$

Teorema 1.2. Sea $f : \mathbb{R}^n \to \mathbb{R}$ una función continuamente diferenciable. Entonces la dirección donde $f(x)$ crece más rápido es $\nabla f(x)$.

Corolario 1.1. Bajo las condiciones del Teorema (1.2), $f(x)$ decrece más rápido en la dirección $-\nabla f(x)$.

Definición 1.1. Una **dirección de descenso** $d \in \mathbb{R}^n$ para $f \in \mathcal{C}^1$ es un vector tal que

$$f(x + td) < f(x)$$

para $t \in (0, T)$. Es decir, permite que el punto x más cerca al mínimo local x^* de la función objetivo $f : \mathbb{R}^n \to \mathbb{R}$.

Teorema 1.3. Si $g(x)^T d < 0$ entonces d es una dirección de descenso.

Observación. La dirección

$$d_k = -g(x_k)$$

es la elección más obvia de una dirección de búsqueda.

1.2 Método BFGS

El método de Newton es una algoritmo iterativo que permite obtener el óptimo x^* de una función 2 veces continuamente diferenciable $f : \mathbb{R}^n \to \mathbb{R}$. Dicho algoritmo es optimización sin restricciones.
A partir de un punto inicial x_0 se genera una secuencia $\{x_k\}$,

$$x_{k+1} = x_k - (\nabla^2 f_k)^{-1} \nabla f_k$$

donde $\nabla^2 f_k = \nabla^2 f(x_k)$ es el Hessiano y $\nabla f_k = \nabla f(x_k)$ es el gradiente.
Para encontrar el nuevo punto x_{k+1} a partir de x_k se define $d = x - x_k$ y se usa la aproximación de Taylor de segundo orden de la función

$$m_k(d) = f_k + f_k d + \frac{1}{2} d^T \nabla^2 f_k d$$

donde el tamaño de paso se obtiene calculando el gradiente, igualidando a cero y resolviendo para p.
Una de la **problemáticas** del método de Newton es que requiere calcular el Hessiano $\nabla^2 f_k$ cual pueder ser muy costoso. Más aún, requiere el cálculo de la inversa del Hessiano,

$$d_k = -(\nabla^2 f_k)^{-1} \nabla f_k.$$

Otro problema es que no se puede garantizar que d_k sea una dirección de descenso, i.e, puede suceder que

$$d_k^T \nabla f_k = -\nabla^T f_k (\nabla^2 f_k)^{-1} \nabla f_k > 0$$

en alguna o varias iteraciones.
Una alternativa de solución, cuando d_k no es una dirección de descenso, es modificar el Hessiano añadiendo una matrix E_k de modo que la nueva matrix $B_k = \nabla^2 f_k + E_k$ sea definida positiva.
La alternativa anterior garantiza que la dirección d_k sea de descenso, sin embargo, requiere del cálculo de Hessiano. Además necesita de un algoritmo que garantice que la nueva matriz B_k sea definida positiva (p.e. Cholesky).
¿Existe alguna otra alternativa?

- Los métodos cuasi-newton construyen un modelo que se basa en medir los cambios del gradiente.

- Solo requieren del cálculo del gradiente, similar al algoritmo de máximo descenso.

- Su comportamiento es superior al algoritmo de máximo descenso.

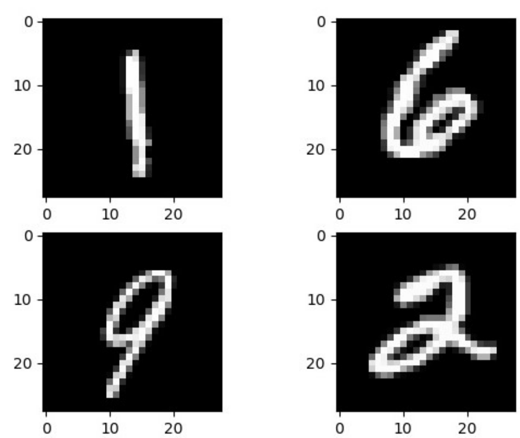
- En lugar de calcular el Hessiano en cada iteración se propone un método que permite calcular una secuencia B_k usando la curvatura medida en el paso actual.

1.3 Datos MNIST

Los datos utilizados corresponden a la base de datos MNIST. Que son imágenes de números dígitos escritos a mano. Las imágenes tiene dimensión 28×28 y la distribución de los datos está dada por

Datos	Cantidad
Entrenamiento	50000
Pruebas	10000
Validación	10000

A continuación se muestra un ejemplo del conjunto de entrenamiento



Dado que la función $F_\theta(\theta)$ corresponde a clasificación binaria, la experimentaición se realizó con las imágenes de etiqueta cero y uno.

2 Método

2.1 Aproximación del Gradiente

Algunas veces puede resultar difícil calcular el gradiente de una función de manera analítica dada su complejidad en cuanto a variables. En vez de hacer eso se puede usar una aproximación de primer orden. Usando expansión de Taylor de primer orden sobre una función $f(x)$ tenemos que

$$f(x+hd) = f(x) + h\nabla^T f(x)d + O(h^2) \quad (1)$$

donde d es un vector unitario, es decir, $||d|| = 1$. Por notivos de notación, llamaremos $\nabla f(x) = g$ y $\nabla^2 f(x) = H$. Así que

$$f(x+hd) = f(x) + hg^T d + O(h^2)$$

Si $d = e_i$, donde e_i denota al vector canónico i, i.e.,

$$e_i^T = \left[0, 0, \ldots, \underbrace{1}_{i}, \ldots, 0\right], i = 1, \ldots, n.$$

Las componentes no nulas correspondientes al i-ésimo componente de e_i quedan como

$$\begin{aligned} f(x+he_i) &= f(x) + hg^T e_i + O(h^2) \\ &= f(x) + hg_i + O(h^2). \end{aligned}$$

reordenando términos tenemos

$$g_i \approx \frac{f(x+he_i) - f(x)}{h}$$

Así obtenemos una **aproxiamción de primer orden** para g. De manera similar podemos aproximar el Hessiano. Sin embargo, para descneso gradiente con paso exacto, estamos interesados en encontrar Hd, esto se puede aproximar mediante

$$g(x+hd) = g + hHd + O(h^2),$$

reordenando términos obtenemos

$$Hd \approx \frac{g(x+hd) - g}{h}$$

2.2 Método BFGS

En el BFGS la idea es calcular la aproximacion de matriz inversa del Hessiano H_{k+1} basado en B_{k+1}. Para ello se considera la ecuación DFP con $\rho_k = \dfrac{1}{s_k y_k}$ y se usa

$$B_{k+1} = (I - \rho_k y_k s_k^T) B_k (I - \rho_k s_k y_k^T) + \rho_k y_k y_k^T$$

y las relaciones

$$\begin{aligned} B_{k+1} s_k &= y_k \\ H_{k+1} y_k &= s_k. \end{aligned}$$

Ahora, $\rho_k = \dfrac{1}{s_k y_k}$ y por lo tanto,

$$H_{k+1} = (I - \rho_k y_k s_k^T) H_k (I - \rho_k y_k y_k^T) + \rho_k s_k y_k^T$$

Así que en cada paso de la actualización se obtiene

1. $d_k = -H_k \nabla f_k$.
2. Calcular α_k usando búsqueda en línea.
3. $x_{k+1} = x_k + \alpha_k d_k$.
4. Calcular ∇f_{k+1}, y_k, s_k, ρ_k y actualizar H_{k+1}.
5. $H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$

2.3 Función para optimizar

Considere el problema de optimización

$$F(\theta) = \frac{1}{N} \sum_{i=1}^{N} (h_\theta(x_i) - y_i)^2$$

donde (x_i, y_i), $x_i \in \mathbb{R}^n$, $y_i = \{0, 1\}$, $i = 1, 2, \ldots, N$ son dados y

$$\begin{aligned} h_\theta(x) &= f_{a,b}(g_{c,d}(x)) \\ g_{c,d} &: \mathbb{R}^n \to \mathbb{R}^m \\ f_{a,b} &: \mathbb{R}^m \to \mathbb{R}. \end{aligned}$$

$m \in \mathbb{N}$ es conocida,

$$\begin{aligned} g_{c,d}(x) &= \left[\sigma(c_j^T x + d_j)\right]_{j=1}^m \\ g_{a,b}(x) &: \sigma(a^T z + b) \\ \sigma(t) &: \frac{1}{1+e^{-t}}, t \in \mathbb{R}. \end{aligned}$$

y θ corresponde al conjunto de parámetros a, b, c, d.

Observaciones

1. La función anterior tiene como variable al conjunto de parámetros θ. Observemos que escribir θ de manera analítica respecto a a, b, c, d no es inmediato.

2. Los parámetros a, b, c, d tienen distintas dimensiones.

3. Encontrar el gradiente explícitamente no es trivial, por lo que se usará una aproxmación de primer oden para la derivada y el gradiente.

2.4 Pseudocódigo

2.4.1 Búsqueda con BackTracking

Input: $\hat{\alpha}$, $\rho \in (0, 1)$, $c_1 \in (0, 1)$
Output: Tamaño de paso α_k
1: Haga $alpha = \hat{\alpha}$
2: $inum = 0$
3: **while** $f(x_k + \alpha d_k) > f(x_k) + c_1 \alpha \nabla f_k^T d_k$ **do**
4: $\quad alpha \to \rho\alpha$
5: **end while**
6: Regresa $\alpha_k = \alpha$

2.4.2 Descenso Gradiente con paso exacto

Input: x_0
Output: x^*
1: Haga $k = 0$
2: **while** $||g_k|| \neq 0$ **do**
3: $\quad d_k = -g_k$
4: $\quad \alpha_k = \dfrac{g_k^T g_k}{g_k^T Q g_k}$
5: $\quad x_{k+1} = x_k + \alpha_k d_k$
6: $\quad k = k + 1$
7: **end while**

2.4.3 Algoritmo BFGS

Input: x_0 y H_0
Output: óptimo x^*
1: $k = 0$
2: **while** $||\nabla f_k|| \neq 0$ **do**
3: $\quad d_k = H_k \nabla f_k$
4: $\quad$ Calcular α_k usando búsqueda en línea.
5: $\quad x_{k+1} = x_k + \alpha_k d_k$
6: $\quad H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$
7: $\quad k = k + 1$
8: **end while**

3 Resultados

Del conjunto de entrenamiento se usaron un total de $10,610$ ejemplos de dígitos cero y uno. θ_0 se generó de manera aleatoria de una distribución normal estándar. Para el algoritmo de búsqueda en línea se utilizaron los parámetros

Parámetro	Valor
α	0.9
ρ	0.5
c_1	10^{-4}
max_{iter}	30

Tanto para descenso gradiente como para BFGS se utilizaron los parámetros generales

Parámetro	max_{iter}	N	n	m	τ_{grad}
Valor	30	10610	784	$\{2, 5\}$	10^{-2}

Para medir el error se usó la función

$$error = \frac{1}{n} \sum_{i=1}^{n} |\mathbf{1}_{h_\theta > 0,5}(x_i) - y_i|$$

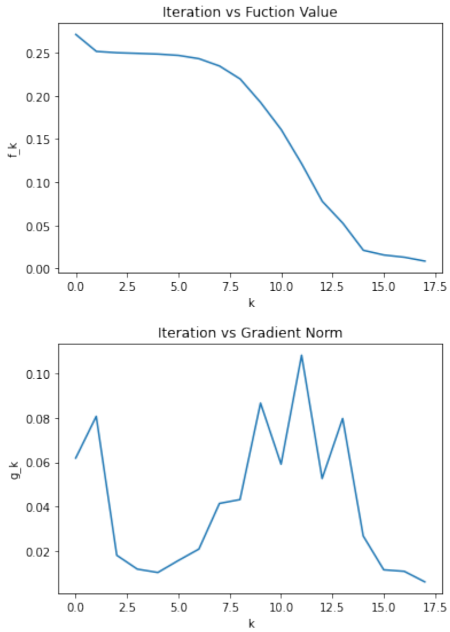
donde $\{(x_i, y_i)\}_{i=1}^n$ se obtiene del conjunto train_set y $x_i \in \mathbb{R}^{784}$ y $y_i \in \{0, 1\}$.

utilizando el conjunto de prueba de $MNIST$.

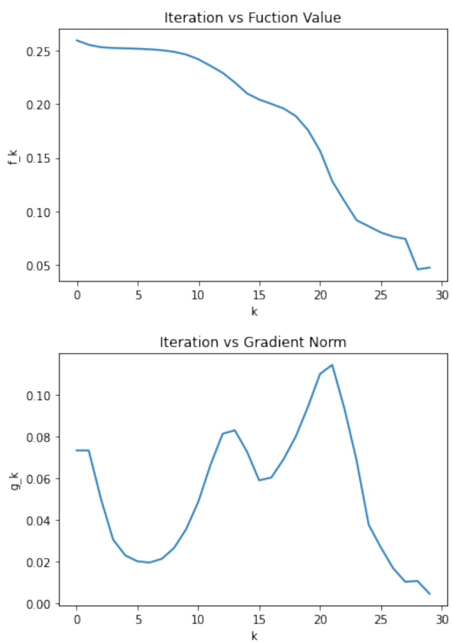
3.1 $m = 2$

Algoritmo	tiempo	iteraciones	error
SD	3848,37 segundos	17	0.0033
BFGS	7962,65 segundos	28	0.0047

3.1.1 Descenso Gradiente



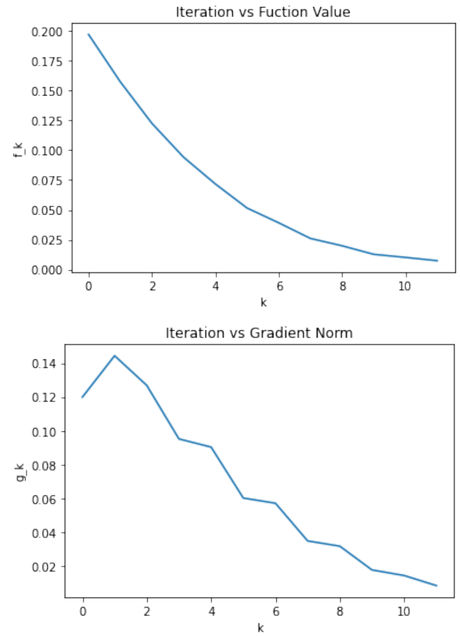
3.1.2 BFGS



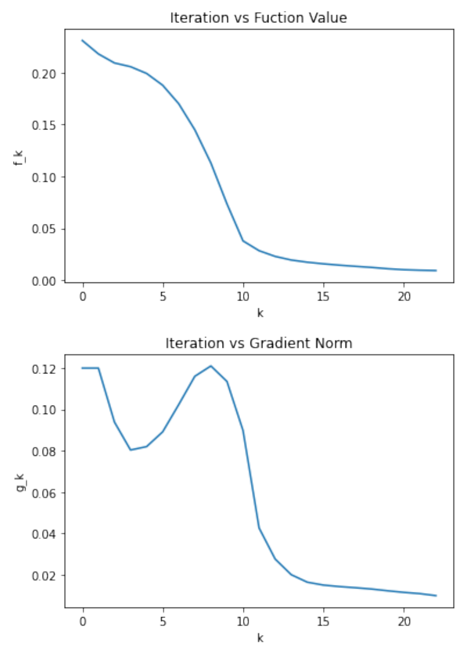
3.2 $m = 5$

Algoritmo	tiempo	iteraciones	error
SD	7345,89 segundos	11	0.0037
BFGS	13974,915 segundos	21	0.0099

3.2.1 Descenso Gradiente



3.2.2 BFGS



4 Conclusiones

- El problema de clasificación binaria de un conjunto de entrenamiento se puede plantear como un problema de optimización. En este caso se usó un conjunto de imágenes de ceros y unos del dataset MNIST.
- En la experimentaicón se observó el costo compu-

tacional de no usar directamente el gradiente sino la aproximación de primer orden. Se observó que una evaluación del gradiente tardó entre 107 y 201 segundos. Es de esperarse que conforme aumente el valor de m, aumente el tiempo de cálculo. Además, en las gráficas se observó mayor estabilidad a mayor valor de m.

- Se observó además que al no obtener directamente la matriz hessiana, el método BFGS es sensible a la inicialización de la matrix H.

- El algoritmo BGFS es presenta un buen desempleño y no requiere el cálculo del Hessiano ni de su inversa.

- La principal limitación del algoritmo BFGS es en problemas donde el número de variable es muy grande (por ejemplo, un millón de variables o más) pues es casi imposible guardar la matriz de tamaño $n \times n$.

- Observemos que el error es pequeño, considerando el número total de imágenes del conjunto de validación. Intuitivamente esto puede ocurrir porque la clasificación es clara para ceros y unos (dado que la forma de escribir estos dígitos es muy diferente). Lo que sugiere probar este algoritmo para clasificar unos y sietes o seis y nueves.