Tarea 5. Optimización, Gradiente Conjugado Lineal

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

***Resumen*—En esta tarea se resuelve un problema de optimización cuadrático para el suavizado de ruido sobre ciertos datos generado de una función, en particular datos son formados con la función $f(x,y) = x^2 + y^2 + r$ donde r es un término de ruido proveniente de una distribución uniforme entre 0 y 1. La solución presentada a este problema se aproxima con el uso del algoritmo de Gradiente Conjundo Lineal.**

***Index Terms*—Gradiente Conjugado**

I. Introduction

El método de Gradiente Conjugado es un método iterativo para solucionar sistemas de ecuaciones lineales (se puede extender al caso no lineal) de la forma $Ax = b$ dónde A es una matriz simétrica definida positiva. Equivalentemente puede ser plateado para resolver el problema minimización cuadrático dado por $min\Phi(x) = \frac{1}{2}x^TAx - b^Tx$. Una de las características de este método es que genera un conjunto de vectores $p_0, p_1, \ldots, p_l$ tal que cada uno de ellos comple la propiedad de ser conjugados de la matriz A, es decir que $p_i^TAp_j = 0$ para todo $i \neq j$. Además este conjunto de vectores son mutuamente ortogonales y linealmente independientes.

Dado que $p_0, p_1, \ldots, p_l$ forman una base entonces la solución al sistema $x*$ puede ser escrito como una combinación lineal de dicha base, es decir, $x* = \sum \alpha_i p_i$. Luego

$$Ax* = b$$

$$\sum \alpha_i Ap_i = b$$

multiplicando por p_k

$$p_k^T \sum \alpha_i Ap_i = p_k^T b$$

dado que son ortogonales

$$\alpha_k p_k^T Ap_k = p_k^T b$$

$$\alpha_k = \frac{p_k^T b}{p_k^T Ap_k}$$

Esto da a una solución $x*$ al sistema en dónde primero calculamos n direcciones conjugadas y luego calculamos los ceficientes α_k. El algoritmo iterativo de Gradiente Conjugado cálcula estas direcciones y coeficientes en cada paso. En el Apéndice se muestra el pseudo código de este.

El problema de optimización cuadrática a resolver es el siguiente:

$$arg\min_x \sum_{i,j} \left[(x_{i,j} - g_{i,j})^2 + \lambda \sum_{(l,m)\in\Omega_{i,j}} (x_{i,j} - x_{l,m})^2 \right] \quad (1)$$

dónde $\lambda > 0$ es un parámetro de regularizacion dado. g es la función la cual se desea suavizar (o filtrar el ruido) con $g(x,y) = x^2 + y^2 + r$ donde r es un término de ruido proveniente de una distribución uniforme entre 0 y 1; ver Fig 1. Y $\Omega_{i,j} = \{(i+1,j),(i-1,j),(i,j+1),(i,j-1)\}$ es el conjunto de índices formados por posiciones vecinas a (i,j).

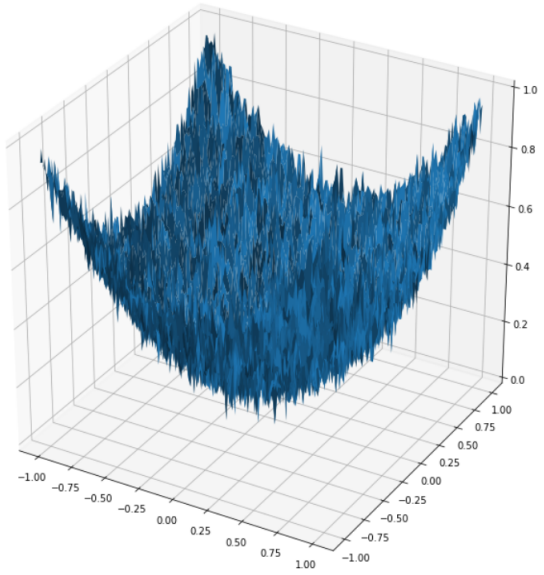

Figura 1. Gráfica de $g(x,y)$ con $x, y \in [0,1]$

II. Métodología

Dado que tenemos que generar valores de nuestra función $g(x,y)$ se fijaron los límites del problema a los rangos $x_i, y_j \in [0,1]$ así generamos una matriz con valores $g_{i,j} = g(x_i, y_j)$ de dimensiones nxm.

Cómo forma de simplifiación se procedió a representar a la matriz g como un solo vector (concatenando por renglones) y a reescribir la ecuación (1) de la siguiente forma:

$$arg\min_x f(x) = \sum_{i=0}^{nxm-1} \left[(x_i - g_i)^2 + \lambda \sum_{(k)\in\Omega_i} (x_i - x_k)^2 \right] \quad (2)$$

con $\Omega_i = \{(i+1), (i-1), (i+m), (i-m)\}$. Notemos que en la representación anterior los vecinos de $x_{i,j}$ estaban localizados en su correspondiente matriz en arriba, abajo, a la izquierda y a la derecha, en esta nueva los vecinos laterales se conservan en el vector pero los vecinos de arriba y abajo ahora están a una distancia $\pm m$ en la nueva representación vectorial. Notemos que las posiciones k en el vector de los vecinos generados deben de estar dentro del rango $[0, nxm)$ y además los vecinos laterales en alguna posición k de un elemento i también deben cumplir que $\lfloor k/m \rfloor = \lfloor i/m \rfloor$, es decir, pertenezcan al mismo renglón.

Derivando (2) con respecto a x_i obtenemos que el gradiente es $\nabla f(x) = [t_1, t_2, \ldots, t_{nxm}]^T$ con

$$t_i = 2(x_i - g_i) + 4\lambda \sum_{k\in\Omega_i} (x_i - x_k) \quad (3)$$

y el Hessiano H tal que

$$H_{i,i} = 2 + 4\lambda(\#\Omega_i)$$

$$H_{i,i+1} = H_{i,i-1} = H_{i,i+m} = H_{i,i-m} = -4\lambda$$

con $\#\Omega_i$ como el numero de vecinos válidos para x_i tal que cumplan las restricciones sobre el vector mencionadas anteriormente.

Una vez obtenida nuestro nueva representación para el problema se procedió a encontrar una solución para valores $\lambda = 1$, $\lambda = 100$ y $\lambda = 1000$, los resultados obtenidos se muestran en la siguiente sección.

Durante la implementación se usó la Q del algoritmo como el Hessiano de la función y para reducir uso de memoria procesamiento, dado que Q puede ser una matriz muy grande en su mayoría con ceros, se implemetó una función que calcula el producto Qd_k en cada iteración.

III. Resultados

Se realizaron varias pruebas con valores $\lambda = 1$, $\lambda = 100$ y $\lambda = 1000$ en la tabla I se muestra una comparativa de los resultados obtenidos.

Cuadro I
Tabla comparativa para distintos valores de λ

λ	***Iteraciones***	$\|\|\nabla g(x*)\|\|$	$g(x*)$
1	50	8.5611e-9	27.2598
100	101	5.9590	209.4859
1000	103	176.67	542.67

Cómo se puede observar con el valor de $\lambda = 1$ se alcanzó minimizar el error descrito por la ecuación (2) mucho mejor que para los otros valores de λ y el error más grande se obtuvo con $\lambda = 1000$.

Sin embargo con $\lambda = 1$ la poderación al promedio de los errores de los puntos vecinos tiene muy poca contribución por ello la aproximación a la función es mucho más precisa y por tanto también el ruido se ve reflejado en esta misma como podemos ver en la Fig 3.

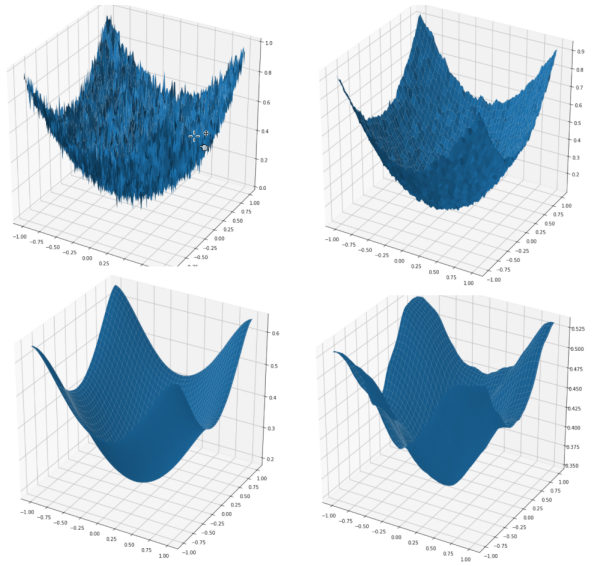

Figura 2. De izquerda a derecha y arriba a abajo; g(x,y), aproximación con $\lambda = 1$, aproximación con $\lambda = 100$ y aproximación con $\lambda = 1000$

Para $\lambda = 100$ se obtuvo un error de aproximación más alto pero como observamos en la figura 4 la gráfica es muy similar a la función original sin ruido, esta vez la contribución del promedio de los errores de los puntos vecinos tiene mayor peso para el calculo del error en un punto actual, por tanto se espera que el error en esa zona sea mejor promediado suavizando la función mucho mejor.

Con $\lambda = 1000$ el peso del error calculado en un punto tiene menor efecto puesto que dependerá en mayor medida de la poderación del promedio del error de los pesos de los vecinos provacando que el suavizado no sea tan *suave* , tal y como se observa en la figura 5.

IV. Conclusiones

Los resultados obtenidos usando el método de Gradiente Conjugado fueron abstante buenos, en este caso la funcion a minimizar era una función cuadrática que calculaba el error el error de aproximación junto con una ponderación del promedio de los errores de los 4 vecinos más cercanos.

Se obtuvo mejores resultados en el savizado usando $\lambda = 100$ tal y como se observa en las figuras 4 y 2, con valore más bajos de lambda la contribución del promediado de los errores de los vecinas no contribuye demasiado haciedo que la aproximación se ajuste demasiado a la función original como se vé en la figura 3 y para valores de α mucho mayores el suavizado es un poco más abrupto tratándose de parecer más a las regiones de vecinos puesto que tienen mucho mayor poderación como se observa en la figura 5.

V. Apéndice

V-A. Algoritmos

Algorithm 1: Algoritmo Gradiente Conjugado

Result: x^*

x_k <- Inicializar

$g_0 = Ax_0 - b$

$d_0 = -g_0$ **while** $||g_k|| > tol$ **do**

$q_k = Ad_k$

$\alpha_k = \frac{g_k^T g_k}{d_k^T q_k} = -\frac{g_k^T d_k}{d_k^T q_k}$

$x_{k+1} = x_k + \alpha_k d_k$

$g_{k+1} = g_k + \alpha_k q_k = \nabla f(x_{k+1})$

$\beta = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} = \frac{g_k^T q_k}{d_k^T q_k}$

$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$

end

V-B. Gráficos complementarios

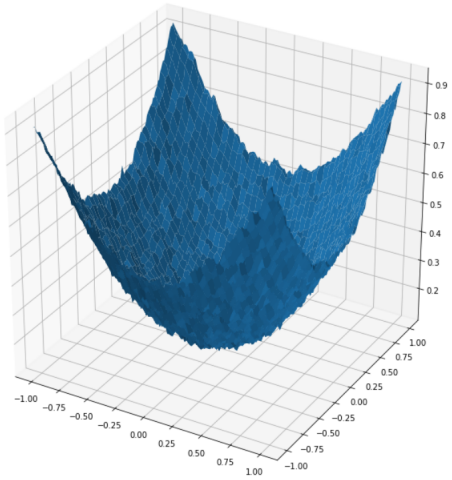

Figura 3. Aproximación a la función con $\lambda = 1$

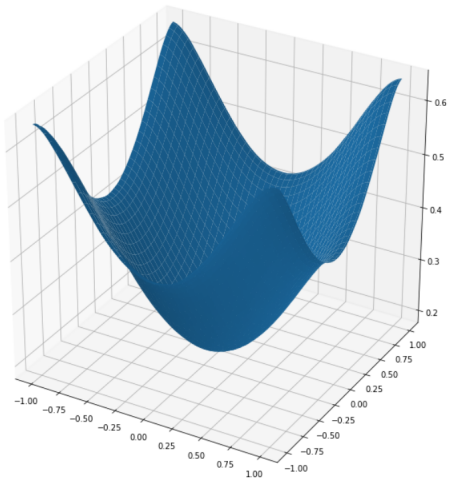

Figura 4. Aproximación a la función con $\lambda = 100$

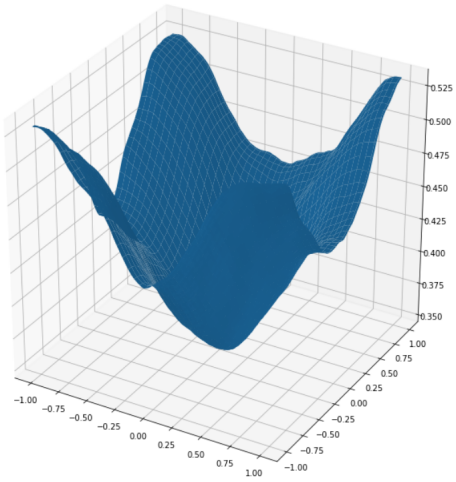

Figura 5. Aproximación a la función con $\lambda = 1000$

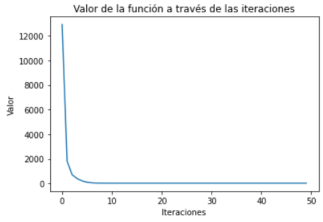

Figura 6. Grafica de la funcion de error a través de las iteraciones con $\lambda = 1$

Referencias

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

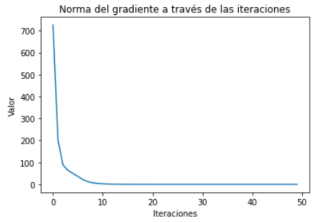

Figura 7. Grafica de gradiente de la funcion de error a través de las iteraciones con $\lambda = 1$

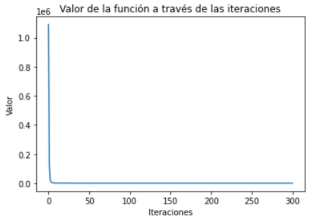

Figura 8. Grafica de la funcion de error a través de las iteraciones con $\lambda = 100$

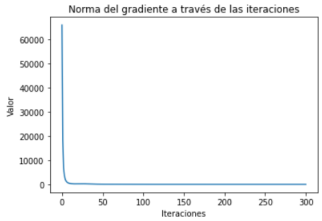

Figura 9. Grafica de gradiente de la funcion de error a través de las iteraciones con $\lambda = 100$

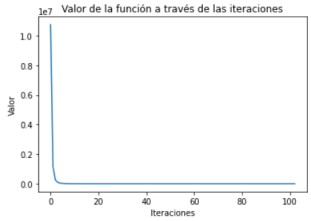

Figura 10. Grafica de la funcion de error a través de las iteraciones con $\lambda = 1000$

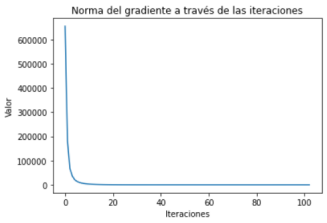

Figura 11. Grafica de gradiente de la funcion de error a través de las iteraciones con $\lambda = 1000$