Tarea 8: Problemas de Mínimos Cuadrados No Lineales

Juan Gerardo Fuentes Almeida

Abstract—En esta práctica se implementan los algoritmos de Gauss-Newton y Levenberg-Maquardt para resolver problemas de optimización con Mínimos Cuadrados No Lineales, aplicados a desenvolvimiento de fases.

1 Introducción

En Problemas de Mínimos Cuadrados, la función objetivo f tiene la forma especial:

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x)$$

donde cada r_j es una función residual suave de $\Re^n$ a $\Re$, esta forma especial hace que el problema sea mas fácil de resolver comparado con un problema general de optimización sin restricciones.

Ensamblamos los componentes individuales de r_j en un vector residual $r : \Re^n \to \Re^m$, de la siguiente manera:

$$r(x) = (r_1(x), r_2(x), ..., r_m(x))^T$$

Con esta notación, podemos reescribir f como $f(x) = \frac{1}{2} \parallel r_j^2(x) \parallel_2^2$, y sus derivadas en términos del Jacobiano $J(x)$:

$$J(x) = \left[\frac{\partial r_j}{\partial x_i}\right]_{\substack{j=1,2,...,m \\ i=1,2,...,n}} = \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_m(x)^T \end{bmatrix}$$

El gradiente y el Hessiano de f se pueden expresar de la siguiente manera:

$$\begin{aligned} \nabla f(x) &= \sum_{j=1}^{m} r_j(x)\nabla r_j(x) = J(x)^T r(x), \\ \nabla^2 f(x) &= \sum_{j=1}^{m} \nabla r_j(x)\nabla r_j(x)^T + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x) \\ &= J(x)^T J(x) + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x). \end{aligned}$$

En muchas aplicaciones, las primeras derivadas parciales de los residuales, y por tanto, de la Matriz Jacobiana $J(x)$ son relativamente fáciles de calcular. Asimismo, utilizando $J(x)$ podemos calcular el primer término $J(x)^T J(x)$ del Hessiano sin evaluar las segundas derivadas de la función $r(x)$, esto es distintivo de los problemas de mínimos cuadrados, ya que los residuos r_j están cerca de la solución, y por tanto, $\nabla^2 r_j(x)$ es relativamente pequeño.

2 Teoría

2.1 Método de Gauss-Newton

Puede verse como un método de Newton modificado con una búsqueda en linea, en el cual se resuelve el siguiente sistema para obtener la dirección de búsqueda p_k:

$$J_k^T J_k p_k^{GN} = -J_k^T r_k$$

Este método simplifica el tiempo de cálculo aproximando $\nabla^2 f_k$ mediante el producto $J_k^T J_k$; éste término dominante mantiene un tiempo de convergencia cercano al del método de Newton. Además, p_k^{GN} es una dirección de descenso si $J(x)$ tiene rango completo y ∇f_k es diferente de cero.

2.2 Método de Levenberg-Maquardt

Este método, al igual que el de Gauss-Newton utiliza la misma aproximación del Hessiano, pero reemplazando la búsqueda en linea con una estrategia de región de confianza, evitando que el sistema tenga un rango deficiente, como puede ocurrir en el método de Gauss-Newton, al mismo tiempo que mantiene también propiedades de convergencia similares. En este método se resuelve un sistema de ecuaciones de la forma:

$$(J_k^T J_k + \lambda I)p_k^{LM} = -J_k^T r_k$$

para un escalar λ, en ésta implementación se propone un valor mayor al eigenvalor más pequeño de la matriz $J_k^T J_k$.

3 IMPLEMENTACIÓN

Sea $f(x)$ la fase desenvuelta de una imagen, y $g(x)$ su fase envuelta, por la propiedad de Nyquist, si la derivada de una función esta acotada en $(-\pi, \pi)$, es decir, $f(x) - f(x - e_1) \in (-\pi, \pi)$, entonces:

$$f(x) - f(y) = W[g(x) - q(y)]$$

siendo W el operador de envolvimiento y $\| y - x \| = 1$

luego, para recuperar $f(x)$ de $g(x)$, calculamos:

$g'_1(x) = g(x) - g(x - e_1)$, $g'_2(x) = g(x) - g(x - e_2)$

y envolvemos la función resultante:

$\widehat{g'_1(x)} = W[g'_1(x)]$, $\widehat{g'_2(x)} = W[g'_2(x)]$

Por tanto, por el criterio de Nyquist, podemos aproximar el residuo entre esta función y la derivada de la fase desenvuelta mediante mínimos cuadrados:

$$U(f) = \sum_{\Omega}[\widehat{g'_1(x)} - f'_1(x)]^2 + \sum_{\Omega}[\widehat{g'_2(x)} - f'_2(x)]^2$$

donde $f'_1(x) = f(x) - f(x - e_1)$ y $f'_2(x) = f(x) - f(x - e_2)$ son las derivadas de la fase desenvuelta, las cuales podemos aproximar mediante Bases de Funciones Radiales:

$$f(x) = \sum_{ij} \alpha_{ij} \phi(i, j, x)$$

con $\phi(i, j, x) = \exp[\frac{-1}{2\sigma_{ij}}(x - \mu_{ij})^2]$

luego

$f'_1(x) = \sum_{ij} \alpha_{ij}[\phi(i, j, x) - \phi(i, j, x - e_1)] = \sum_{ij} \alpha_{ij} \varphi_1(i, j, x)$

y $f'_2(x) = \sum_{ij} \alpha_{ij}[\phi(i, j, x) - \phi(i, j, x - e_2)] = \sum_{ij} \alpha_{ij} \varphi_2(i, j, x)$

luego, para recuperar $f(x)$ de $g(x)$, tendremos que minimizar la siguiente expresión:

$$U(f) = \sum_{\Omega}[\widehat{g'_1(x)} - f'_1(x)]^2 + \sum_{\Omega}[\widehat{g'_2(x)} - f'_2(x)]^2$$

3.1 Optimización de α

Para optimizar el valor de α, el cual es la amplitud de las funciones radiales, se construye un Jacobiano cuyas filas son los componentes en x y en y de la función objetivo (2 veces el numero de pixeles de la imagen)

$$J(\alpha) = \begin{pmatrix} \frac{\partial r(x_1)}{\partial \alpha_1} & \frac{\partial r(x_1)}{\partial \alpha_2} & \cdots & \frac{\partial r(x_1)}{\partial \alpha_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(x_N)}{\partial \alpha_1} & \frac{\partial r(y_N)}{\partial \alpha_2} & \cdots & \frac{\partial r(x_N)}{\partial \alpha_m} \\ \frac{\partial r(y_1)}{\partial \alpha_1} & \frac{\partial r(y_1)}{\partial \alpha_2} & \cdots & \frac{\partial r(y_1)}{\partial \alpha_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(y_N)}{\partial \alpha_1} & \frac{\partial r(y_N)}{\partial \alpha_2} & \cdots & \frac{\partial r(y_N)}{\partial \alpha_m} \end{pmatrix}$$

El gradiente de la función esta determinado por $J_k^T r_k$.

Cada elemento de esta matriz esta definido por la siguiente expresión:

$$\frac{\partial r(x_k)}{\partial \alpha_{ij}} = -\varphi_1(i, j, x_k) = -\frac{(x_{k_1} - \mu_{ij_1})}{\sigma_{ij}^2} \phi(i, j, x_k)$$

$$\frac{\partial r(y_k)}{\partial \alpha_{ij}} = -\varphi_2(i, j, x_k) = -\frac{(x_{k_2} - \mu_{ij_2})}{\sigma_{ij}^2} \phi(i, j, x_k)$$

3.2 Optimización de μ

Para optimizar el valor de μ, el cual es la media de las funciones radiales, se construye un Jacobiano cuyas filas son los componentes en x y en y de la función objetivo (2 veces el numero de pixeles de la imagen), y cuyas columnas están repartidas entre las componentes en x y en y de la medias (2 veces el numero de funciones radiales):

$$J(\mu) = \begin{pmatrix} \frac{\partial r(x_1)}{\partial \mu(x_1)} & \cdots & \frac{\partial r(x_1)}{\partial \mu(x_m)} & \frac{\partial r(x_1)}{\partial \mu(y_1)} & \cdots & \frac{\partial r(x_1)}{\partial \mu(x_m)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(x_N)}{\partial \mu(x_1)} & \cdots & \frac{\partial r(x_N)}{\partial \mu(x_m)} & \frac{\partial r(x_N)}{\partial \mu(y_1)} & \cdots & \frac{\partial r(x_N)}{\partial \mu(x_m)} \\ \frac{\partial r(y_1)}{\partial \mu(x_1)} & \cdots & \frac{\partial r(y_1)}{\partial \mu(x_m)} & \frac{\partial r(y_1)}{\partial \mu(y_1)} & \cdots & \frac{\partial r(y_1)}{\partial \mu(x_m)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(y_N)}{\partial \mu(x_1)} & \cdots & \frac{\partial r(y_N)}{\partial \mu(x_m)} & \frac{\partial r(y_N)}{\partial \mu(y_1)} & \cdots & \frac{\partial r(y_N)}{\partial \mu(x_m)} \end{pmatrix}$$

El gradiente de la función esta determinado por $J_k^T r_k$.

Cada elemento de esta matriz esta definido por la siguiente expresión:

$$\frac{\partial r(x_k)}{\partial \mu(x_{ij})} = -\alpha_{ij} \frac{\partial \varphi_1(i,j,x_k)}{\partial \mu(x_{ij})} = -\frac{\alpha_{ij}}{\sigma_{ij}^2} \phi(i, j, x_k)[1 - \frac{(x_{k_1} - \mu_{ij_1})^2}{\sigma_{ij}^2}]$$

$$\frac{\partial r(x_k)}{\partial \mu(y_{ij})} = \frac{\alpha_{ij}}{\sigma_{ij}^4} \phi(i, j, x_k)(x_{k_1} - \mu_{ij_1})(x_{k_2} - \mu_{ij_2})$$

$$\frac{\partial r(y_k)}{\partial \mu(x_{ij})} = \frac{\alpha_{ij}}{\sigma_{ij}^4} \phi(i, j, x_k)(x_{k_1} - \mu_{ij_1})(x_{k_2} - \mu_{ij_2})$$

$$\frac{\partial r(y_k)}{\partial \mu(y_{ij})} = -\alpha_{ij} \frac{\partial \varphi_1(i,j,x_k)}{\partial \mu(y_{ij})} = -\frac{\alpha_{ij}}{\sigma_{ij}^2} \phi(i, j, x_k)[1 - \frac{(x_{k_2} - \mu_{ij_2})^2}{\sigma_{ij}^2}]$$

3.3 Optimización de σ

Para optimizar el valor de σ, el cual es la desviación estándar de las funciones radiales, se construye un Jacobiano cuyas filas son los componentes en x y en y de la función objetivo (2 veces el numero de pixeles de la imagen)

$$J(\alpha) = \begin{pmatrix} \frac{\partial r(x_1)}{\partial \sigma_1} & \frac{\partial r(x_1)}{\partial \sigma_2} & \cdots & \frac{\partial r(x_1)}{\partial \sigma_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(x_N)}{\partial \sigma_1} & \frac{\partial r(y_N)}{\partial \sigma_2} & \cdots & \frac{\partial r(x_N)}{\partial \sigma_m} \\ \frac{\partial r(y_1)}{\partial \sigma_1} & \frac{\partial r(y_1)}{\partial \sigma_2} & \cdots & \frac{\partial r(y_1)}{\partial \sigma_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(y_N)}{\partial \sigma_1} & \frac{\partial r(y_N)}{\partial \sigma_2} & \cdots & \frac{\partial r(y_N)}{\partial \sigma_m} \end{pmatrix}$$

El gradiente de la función esta determinado por $J_k^T r_k$.

Cada elemento de esta matriz esta definido por la siguiente expresión:

$$\frac{\partial r(x_k)}{\partial \sigma_{ij}} = -\frac{\alpha_{ij}}{\sigma_{ij}^3}(x_{k_1} - \mu_{ij_1})\phi(i,j,x_k)[2 - \frac{(x_k-\mu_{ij})^2}{\sigma_{ij}}]$$

$$\frac{\partial r(y_k)}{\partial \sigma_{ij}} = -\frac{\alpha_{ij}}{\sigma_{ij}^3}(x_{k_2} - \mu_{ij_2})\phi(i,j,x_k)[2 - \frac{(x_k-\mu_{ij})^2}{\sigma_{ij}}]$$

4 Resultados

A continuación se muestran las imágenes que se obtuvieron durante la ejecución de la práctica, para diferente número de funciones radiales en una imagen de prueba, utilizando los métodos de Gauss-Newton y Levenberg-Maquardt:

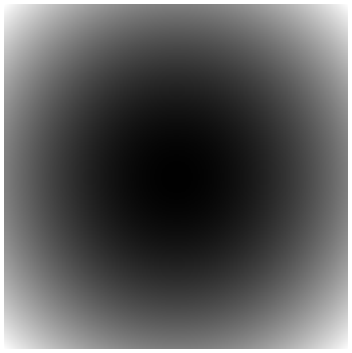
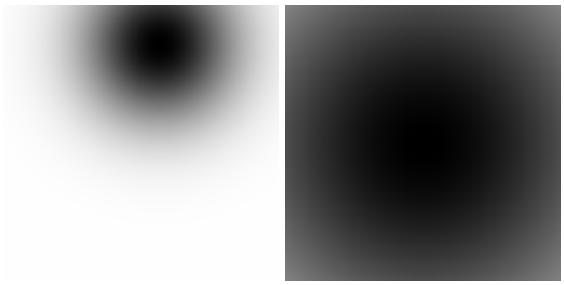
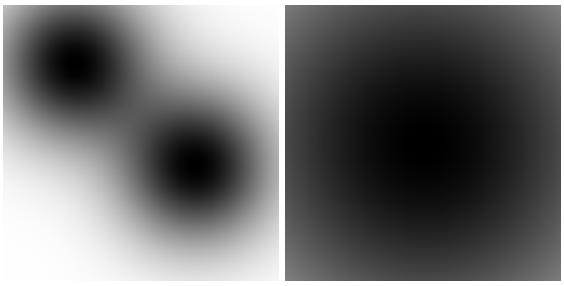


Imagen Desenvuelta

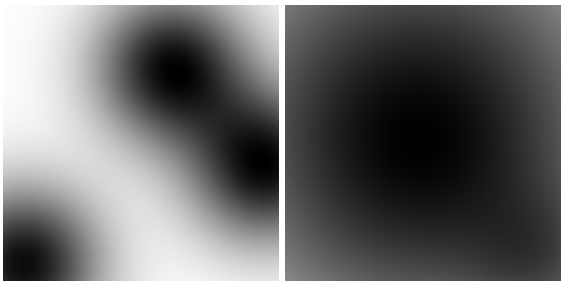
Los siguientes resultados fueron obtenidos con el método de Gauss-Newton:



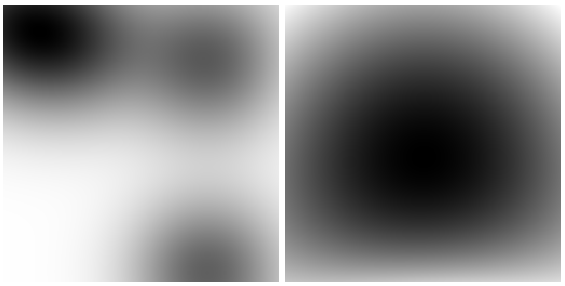
Der. a izq.: Imagen Inicial con 1 RBF, Después de 20 iteraciones



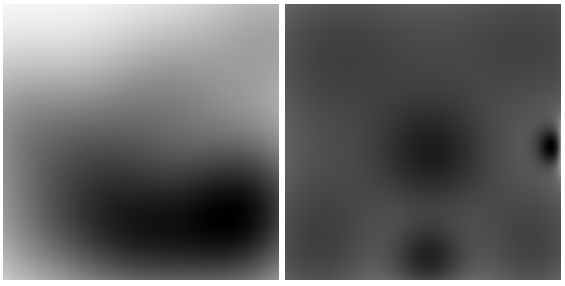
Der. a izq.: Imagen Inicial con 2 RBF, Después de 20 iteraciones



Der. a izq.: Imagen Inicial con 3 RBF, Después de 20 iteraciones



Der. a izq.: Imagen Inicial con 4 RBF, Después de 20 iteraciones



Der. a izq.: Imagen Inicial con 10 RBF, Después de 20 iteraciones

La siguiente tabla muestra el desempeño de los algoritmos implementados en esta práctica, para diferente numero de RBF's, como se observa, el comportamiento de ambos algoritmos es muy similar:

Método	No. de RBF	Iteraciones	Tiempo	Función Objetivo
G-N	1	20	3.464	26003.147407
G-N	2	20	6.25	26003.165339
G-N	3	20	9.149	26002.656034
G-N	4	20	12.481	26003.165815
G-N	10	20	31.329	25915.891924
L-M	1	20	3.943	26003.260303
L-M	2	20	6.219	25986.572096
L-M	3	20	9.246	26002.072557
L-M	4	20	12.125	26001.110397
L-M	10	20	32.809	25898.619399

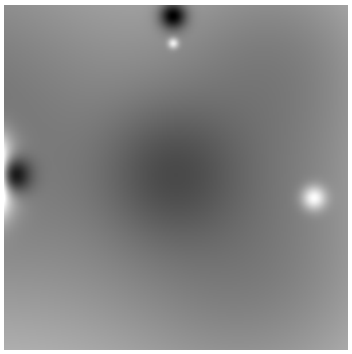
Desempeño de los métodos Gauss-Newton (G-N) y Levenberg-Maquardt (L-M)

5 Conclusiones

Cabe señalar que como el inicio del algoritmo es estocástico, conviene correr el programa varias veces para obtener un

resultado deseable, y como es de esperarse, cuantas más señales gaussianas haya, le cuesta más trabajo al algoritmo optimizar la función objetivo, aunque se observa que el algoritmo sigue un patrón: se coloca una gaussiana en el centro y las demás en los bordes, comúnmente en las esquinas y sobre el punto medio de los lados, después de esto, se procede a disminuir el tamaño de las gaussianas en los bordes y a aumentar el tamaño de la gaussiana en el centro, basta con correr el algoritmo varias veces para un número grande de gaussianas (aprox. 10) para apreciar este comportamiento.

Otro detalle que se observó, fue que en ocasiones el resultado se asemeja al que se muestra en la siguiente figura:



Este resultado no necesariamente es erróneo, ya que debemos considerar que la normalización de la imagen altera la percepción de las intensidades, en esta imagen se aprecia que los puntos muy negros de la parte superior y de la izquierda tienen impacto sobre la normalización de los pixeles de la imagen, considerando también que el resultado de la función objetivo para este caso fue similar a los otros resultados mostrados en las imágenes anteriores; la mancha semi-oscura del centro muestra que el algoritmo si trató de realizar la optimización, aunque la parte de la optimización de las medias no fue suficientemente eficiente para desplazar las señales indeseables hacia afuera de la imagen.

REFERENCES

[1] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

APPENDIX

El programa está implementado tomando en cuenta todas estandarizaciones indicadas en el curso.

El *makefile* que se ha generado, incluye los comandos *make*, *run*, *test* y *clean*. El programa recibe el nombre del archivo de la imagen y el parámetro *methodID*, el cual indica al programa que método de optimización se va a usar, 1 para Gauss-Newton y 2 para Levenberg-Maquardt, el comando *test* ejecuta 20 iteraciones del método de Gauss-Newton con 1 gaussiana.

```
almeida@almeida-X501A1: ~/Code/ej1
almeida@almeida-X501A1:~/Code/framework$ make
gcc -std=c++0x -lstdc++ -lfftw3 -lfftw3f -I include -c src/MatrixInterface.cpp
-o obj/MatrixInterface.o
gcc -std=c++0x -lstdc++ -lfftw3 -lfftw3f -I include -c src/PGMFunctions.cpp -o
obj/PGMFunctions.o
ar  rcs obj/libframework.a  obj/MatrixInterface.o  obj/PGMFunctions.o
almeida@almeida-X501A1:~/Code/framework$ cd
almeida@almeida-X501A1:~$ cd /home/almeida/Code/ej1
almeida@almeida-X501A1:~/Code/ej1$ make
g++ -std=c++0x -I ../framework/include -I ../framework/algorithms -I . -lstdc++
 -lfftw3 -lfftw3f -c main.cpp -o main.o
g++ -o main  main.o -L ../framework/obj -lframework -lfftw3
almeida@almeida-X501A1:~/Code/ej1$ make test
./main TestA/wrappingN.pgm 1
26371.433520
almeida@almeida-X501A1:~/Code/ej1$
```