Tarea 10 - Optimización

Erick Salvador Alvarez Valencia

CIMAT A.C.,
erick.alvarez@cimat.mx

Resumen En el presente reporte se hablará sobre la implementación del método de Levenberg-Marquardt en dos ejercicios, el primero fue ajustar un modelo que correspondía a la función de Rosenbrock en 2 dimensionesm el cual se escribió de una forma vectorial. Para este ejercicio se mostrarán los resultados aplicando diferentes valores de v.
El otro ejercicio consistió en ajustar un modelo no lineal para un conjunto de observaciones en donde se tenía el caso de $m > n$, es decir, existían más observaciones que parámetros. Para este ejercicio también se escribió la función de error en una versión vectorial y al final se generó la gráfica de aproximación basada en el modelo no lineal. Al final del documento se presentarán algunas conclusiones del tema. Finalmente se darán algunas conclusiones.

Keywords: Método LM, Optimización de funciones, Mínimos cuadrados no lineales.

1. Método Levenberg-Marquardt

Lo primero que se realizó fue una implementación del método de Levenberg-Marquardt el cual se basa en optimizar una función de costo escrita de la siguiente manera:

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

Donde $R(x)$ esta definida como

$$R(x) = \begin{pmatrix} r_1(x) \\ r_2(x) \\ \vdots \\ r_m(x) \end{pmatrix} \tag{2}$$

El método LM se basa es resolver un problema de región de confianza para encontrar el siguiente punto de la sucesión.

$$\begin{aligned} min_{x+}||R(x_c) + J(x_c)(x_+ - x_c)||_2 \\ sujeto\ a\ ||x_+ - x_c||_2 \leq \delta \end{aligned} \tag{3}$$

La solución del problema anterior es:

$$(J(x_c)^T J(x_c) + \lambda I)p^{LM} = -J(x_c)^T R(x_c) \tag{4}$$

si y sólo si p^{LM} es factible y existe una constante $\lambda \geq 0$ tal que

$$\lambda(\delta - ||p^{LM}||) = 0 \tag{5}$$

donde $J(x)$ de la ecuación (3) es la matriz Jacobiana de la función $R(x)$.
Como se puede apreciar, para encontrar el siguiente punto de la sucesión p^{LM} se tiene que resolver un sistema de ecuaciones en cada iteración del algoritmo.
Como es de esperarse de este tipo de algoritmos que optimizan un modelo, hay que tener control sobre el radio de la región de confianza del mismo. Por lo cual en cada iteración debemos ver si el modelo se ajustó correctamente a la función y de ser o no ser así reescalamos el valor de λ por un factor v. Para terminar el algoritmo podemos meter una condición de tolerancia en el gradiente de la función el cual está definido como $\nabla f(x_k) = J^T(x_k)R(x_k)$. O en su defecto determinar un máximo número de iteraciones del algoritmo.

2. Ejecuciones y resultados

Para esta tarea se hicieron dos ejercicios los cuales se explican a continuación, para cada uno de ellos se incluirán los resultados obtenidos.

2.1. Ejercicio 1

En este primer ejercicio se hizo la implementación del algoritmo LM y se probó con la función de Rosenbrock en su versión de 2 dimensiones y 50 dimensiones.
Para hacer esto anterior se definió la función en su forma vectorial, la cual se muestra a continuación:

$$\begin{aligned} r_{2i-1}(x) = 10(x_{2i} - x_{2i-1}^2) \\ r_{2i}(x) = 1 - x_{2i-1} \end{aligned} \tag{6}$$

El Jacobiano de la función es el siguiente:

$$J(x) = \begin{pmatrix} -20x_1 & 10 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -20x_3 & 10 & \cdots & 0 & 0 \\ 0 & 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & & \\ 0 & 0 & 0 & 0 & \cdots & -20x_m & 10 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 0 \end{pmatrix} \tag{7}$$

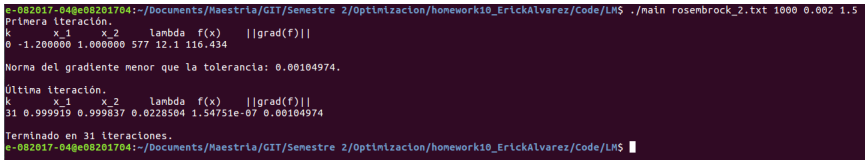
Para cada una de las dos versiones del problema se hicieron pruebas con diferentes valores de v para ver el cambio en la convergencia del algoritmo. A continuación se muestra una tabla con los resultados obtenidos de acuerdo a la descripción de la tarea.
Antes de eso hay que mencionar que como límites se puso una tolerancia para la norma del gradiente de $\tau = 10^{-3}n$, un límite de 1000 iteraciones y el vector inicial para ambos casos fue $x_0 = (-1,2,1,-1,2,1,...,-1,2,1)^T$.

Cuadro 1: Resultados de las pruebas realizadas al método LM con varios valores de v

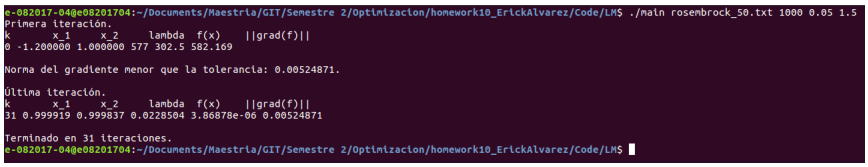
n	v	**Iteración final** k	$(x_k)_1$	$(x_k)_2$	$f(x_k)$	$\|\|\nabla f(x_k)\|\|$
2	**1.25**	40	0.999661	0.999319	5.50138e-07	0.00082822
2	**1.50**	31	0.999919	0.999837	1.54751e-07	0.00104974
2	**2.50**	26	0.999998	0.999996	5.08801e-09	0.00076248
2	**10.0**	39	0.998954	0.997901	1.60179e-11	0.000110417
50	**1.25**	41	0.999906	0.999811	9.82537e-05	0.016867
50	**1.50**	31	0.999919	0.999837	3.86878e-06	0.00524871
50	**2.50**	26	0.999998	0.999996	1.272e-07	0.0038124
50	**10.0**	46	1.000000	1.000000	4.00447e-10	0.000552085

En la Tabla anterior se muestra información sobre lo que ocurrió en la última iteración del algoritmo, entre la cual tenemos: El valor de v, el número de la última iteración, las primeras dos componentes del vector x^*, el valor de la función con el vector anterior mencionado y la norma del gradiente en el mismo vector.
Lo que podemos notar es que para ambas versiones del problema se tuvo convergencia en muy pocos pasos, esto porque el punto inicial estaba relativamente cerca del óptimo. Otra observación que se hace es que vemos que para los primeros valores de v se iba reduciendo el número de iteraciones que realizaba el algoritmo al ir aumentando v excepto para el último valor que era 10 lo cual produjo una convergencia en muchas más iteraciones de las que ya se tenía.
Finalmente se puede observar que las columnas 4, 5 y 6 las cuales corresponden a las dos primeras componentes del vector x^* y a la función evaluada en este vector coinciden con lo esperado, ya que el óptimo de la función de Rosenbrock se

encuentra en el vector $x^{opt} = (1, 1, ..., 1)^T$ y el valor en ese vector es $f(x^{opt}) = 0$. A continuación se muestran dos capturas de pantalla las cuales fueron hechas en las últimas iteraciones del algoritmo con las dimensiones 2 y 50 y usando $v = 1{,}5$.



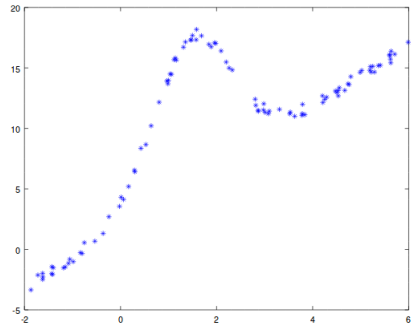
(a) Figura 1. Ejecución del algoritmo LM con dimensión 2 y $v = 1{,}5$.



(b) Figura 2. Ejecución del algoritmo LM con dimensión 50. y $v = 1{,}5$

2.2. Ejercicio 2

En este ejercicio se usó la misma implementación que se realizó en el ejercicio anterior pero esta vez para ajustar un modelo no lineal a un conjunto de puntos provenientes de una función desconocida los cuales se ven de la siguiente forma.



(c) Figura 3. Conjunto de puntos a ajustar

El modelo que ajustaría el conjunto de puntos está definido como:

$$r_i(p) = p_1 x_i + p_2 + p_3 \exp(p_4(x_i - p_5)^2) - y_i \tag{8}$$

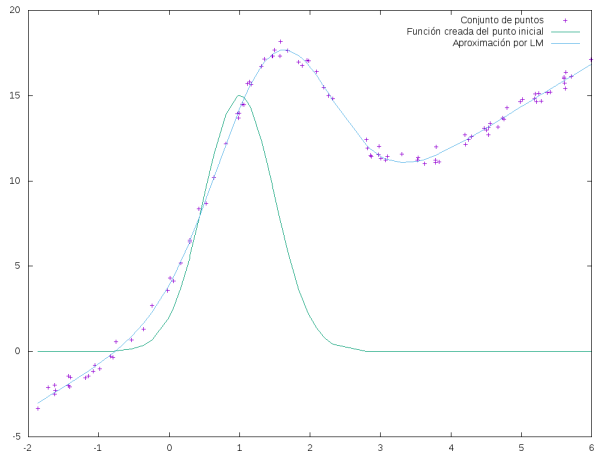
Para $i = 1, 2, ..., m$. Donde x_i, y_i son los datos que ya se conocen y el vector p es el que tiene que encontrar el algoritmo para que el modelo definido en (8) se ajuste al conjunto de puntos.
El Jacobiano de la función es el siguiente:

$$J(x) = \begin{pmatrix} x_1 & 1 & \exp(p_4(x_1 - p_5)^2) & p_3 \exp(p_4(x_1 - p_5)^2)(x_1 - p_5)^2 & -2p_3p_4 \exp(p_4(x_1 - p_5)^2)(x_1 - p_5) \\ x_2 & 1 & \exp(p_4(x_2 - p_5)^2) & p_3 \exp(p_4(x_2 - p_5)^2)(x_2 - p_5)^2 & -2p_3p_4 \exp(p_4(x_2 - p_5)^2)(x_2 - p_5) \\ & & & \vdots & \\ x_m & 1 & \exp(p_4(x_m - p_5)^2) & p_3 \exp(p_4(x_m - p_5)^2)(x_m - p_5)^2 & -2p_3p_4 \exp(p_4(x_m - p_5)^2)(x_m - p_5) \end{pmatrix} \tag{9}$$

Posteriormente se aplicó el algoritmo LM para encontrar el vector p que optimizaba el modelo y así generar la gráfica de la aproximación. Los parámetros usados de entrada al algoritmo son:

1. **Tolerancia** : 0.001m, donde $m = 100$.
2. **Límite de iteraciones** : 1000.
3. **Vector inicial** : $x_0 = (0, 0, 15, -2, 1)^T$.
4. **v** : 1.50.

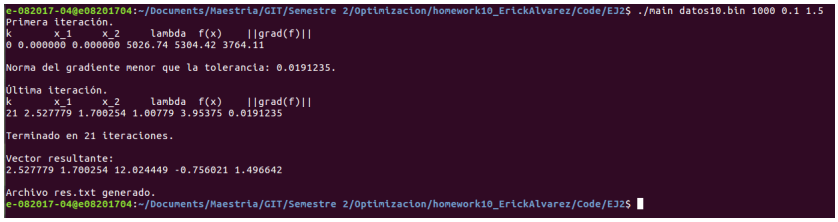
A continuación se muestra la gráfica generada con el conjunto de puntos y la solución encontrada.



(d) Figura 4. Ajuste por LM al conjunto de puntos.

En la Figura anterior podemos ver la gráfica de la función con el punto inicial x_0 en color verde, esta corresponde al vector indicado en los parámetros

anteriores. En morado vemos el conjunto de puntos previamente mostrados y en azul vemos la gráfica generada por el modelo optimizado por mínimos cuadrados no lineales. Podemos notar que el ajuste en dicha gráfica fue muy bueno con respecto al conjunto de puntos.
A continuación se muestra una captura de pantalla de lo que mostró en consola el algoritmo.



(e) Figura 5. Resultado del algoritmo LM para el modelo no lineal.

Se puede ver que el algoritmo realizó 21 iteraciones para poder llegar a la tolerancia pedida.

2.3. Ejercicio 3

La propuesta del proyecto se enviará en un correo a parte a las direcciones de correo indicadas.

3. Notas y comentarios finales.

En el presente reporte se analizó la implementación del método LM para problemas de mínimos cuadrados no lineales, se pudo ver que las implementaciones realizadas ofrecieron buenos resultados a los dos ejercicios aplicados. Además de que la convergencia ofrecida por los métodos fue rápida, esto porque los puntos iniciales dados estaban relativamente cerca del óptimo local.
También se pudo ver el efecto de alterar la v en donde los primeros parámetros hicieron que el método fuera haciendo menos iteraciones, pero cuando se probó con $v = 10$ el método tardó más iteraciones en converger, esto se debe al hecho de que si el modelo ajusta muy bien a la función el algoritmo aumenta el radio de la región de confianza v veces, y como v es muy grande es muy probable que en la siguiente iteración, el ajuste no sea bueno porque el radio creció mucho y como en esa iteración no se actualiza el vector x_k al final se harán más iteraciones para lograr la convergencia.