Optimización Tarea 10

Francisco Javier Peralta Ramírez

6 de mayo de 2018

Resumen

En está tarea se programa el algoritmo *Levenberg-Marquardt* para resolver *Mínimos Cuadrados No Lineales* y se prueba resolviendo la función de Rosenbrock y con un conjunto de puntos dados.

1. Introducción

El método de *Gauss-Newton* es como el método de Newton con busqueda en linea pero con una aproximación para la Hessiana, el método de *Levenberg-Marquardt* se puede obtener con la misma aproximación para la Hessiana pero remplanzando la busqueda en linea por una estrategia de región de confianza. Esto evita una de las debilidades de *Gauss-Newton*, su comportamiento cuando el Jacobiano (J) no es de rango completo.

El problema está dado por

$$\min_p \frac{1}{2}||Jp + r||^2, \qquad \text{sujeto a} \quad ||p|| < \delta$$

donde $\delta > 0$ es el radio de la región de confianza y sólo si existe una constante $\lambda \geq 0$ tal que

$$\begin{aligned}(J^\intercal J + \lambda I)p &= -J^\intercal r \\ \lambda(\delta - ||p||) &= 0\end{aligned}$$

Cuando p está dentro de la región de confianza, $||p|| < \delta$, $\lambda = 0$ y por lo tanto p coincide con *Gauss-Newton*. Por otra parte sí λ es muy grande $J^\intercal J + \lambda I \approx \lambda I$ por lo que

$$\begin{aligned}p &= -\frac{1}{\lambda}J^\intercal r \\ &= -\frac{1}{\lambda}\nabla f(\boldsymbol{x})\lambda(\delta - ||p||)\end{aligned}$$

por lo tanto se comporta como descenso de gradiente.

2. Levenberg-Marquardt

El algoritmo *Levenberg-Marquardt* es:

Algorithm 1 LM

function LM(x_0, R, J, t, i)
  r := $R(x_0)$
  j := $J(x_0)$
  λ := max_diag($j^\intercal j$)
  k := 0
  while $||j^\intercal r|| > t$ **and** $k < i$ **do**
    $p \leftarrow$ Solve $(j^\intercal j + \lambda I)p = -j^\intercal r$
    $x_{k+1} = x_k + p$
    if $R(x_{k+1})^\intercal R(x_{k+1}) \geq R(x_k)^\intercal R(x_k)$ **then**
      $x_{k+1} = x_k$
      $\lambda = \nu\lambda$
    else
      $\lambda = \lambda/\nu$
    end if
    k := $k + 1$
  end while
  return x_k
end function

3. Resultados

Para probar nuestro algoritmo se usó la función Rosenbrock en residuales la cual está dada por

$$\begin{aligned}r_{2i}(\boldsymbol{x}) &= 10(x_{2i+1} - x_{2i}^2) \\ r_{2i+1}(\boldsymbol{x}) &= 1 - x_{2i}\end{aligned}$$

para $i = 0, 1, \ldots, m-1$
Calculamos el Jacobiano:

$$\begin{aligned}J_{2i,2i}(\boldsymbol{x}) &= -20x_{2i} \\ J_{2i+1,2i}(\boldsymbol{x}) &= -1 \\ J_{2i,2i+1}(\boldsymbol{x}) &= 10\end{aligned}$$

Usando una tolerancia $\tau = 0.001n$ y cambiando ν obtenemos los siguientes resultados

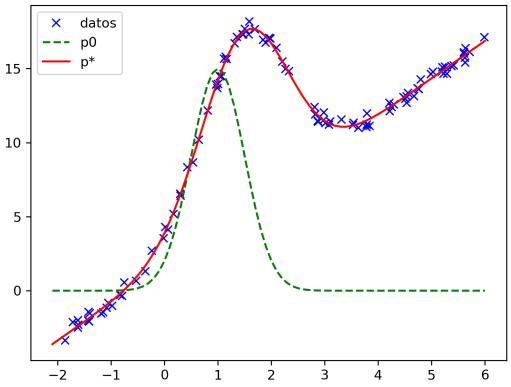
n	ν	k	$(x_k)_1$	$(x_k)_2$	$f(\boldsymbol{x}_k)$	$\|\|\nabla f(\boldsymbol{x}_k)\|\|$
2	1.25	9206	0.996	0.992	7E-6	0.0018
2	1.50	9123	0.996	0.991	9E-6	0.0020
2	2.50	8095	0.996	0.991	9E-6	0.0020
2	10.0	10719	0.996	0.992	7E-9	0.0020
50	1.25	3914	0.941	0.887	0.003	0.04
50	1.50	3571	0.929	0.863	0.005	0.05
50	2.50	3117	0.926	0.857	0.005	0.05
50	10.0	4217	0.940	0.884	0.003	0.05

También probamos ajustando un modelo dado por

$$p_1 x + p_2 + p_3 \exp[p_4 (x - p_5)^2]$$

para el conjunto de datos $(x_i, y_i)_{i=0}^{m-1}$.

Se usó una tolerancia $\tau = 0.001n$, $\nu = 1.25$ y un punto inicial $p_0 = (0, 0, 15, -2.0)^\intercal$. El algoritmo convergió en $k = 823$.



con $p^* = (2.52803, 1.69816, 12.0305, -0.75611, 1.49662)$

4. Conclución

Para el problema de Rosenbrock con $n = 2$ notamos que el desempeño del algoritmo es poco óptimo, a diferencia de métodos como *BFGS*, este tomó muchas iteraciones para encontrar el óptimo, pero aún así tardó menos que una busqueda por descenso de gradiente. Por otra parte cuando se usó para ajustar un modelo no lineal a un conjunto de datos obtuvimos buenos resultados.