optim_tarea07

March 19, 2022

1 Curso de Optimización (DEMAT)

1.1 Tarea 7

Descripción:	Fechas
Fecha de publicación del documento:	**Marzo 19, 2022**
Fecha límite de entrega de la tarea:	**Marzo 27, 2022**

1.1.1 Indicaciones

- Envie el notebook que contenga los códigos y las pruebas realizadas de cada ejercicio.
- Si se requiren algunos scripts adicionales para poder reproducir las pruebas, agreguelos en un ZIP junto con el notebook.
- Genere un PDF del notebook y envielo por separado.

1.2 Ejercicio 1 (5 puntos)

Programar el método de Gauss-Newton para resolver el problema de mínimos cuadrados no lineales

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

donde $r_j : \mathbb{R}^n \to \mathbb{R}$ para $j = 1, ..., m$. Si definimos la función $R : \mathbb{R}^n \to \mathbb{R}^m$ como

$$R(z) = \begin{pmatrix} r_1(z) \\ \vdots \\ r_m(z) \end{pmatrix},$$

entonces

$$min_z f(z) = \frac{1}{2} R(z)^\top R(z).$$

Dar la función de residuales $R(z)$, la función Jacobiana $J(z)$, un punto inicial z_0, un número máximo de iteraciones N, y una tolerancia $\tau > 0$.

1. Hacer $res = 0$.
2. Para $k = 0, 1, ..., N$:

- Calcular $R_k = R(z_k)$
- Calcular $J_k = J(z_k)$
- Calcular la dirección de descenso p_k resolviendo el sistema

$$J_k^\top J_k p_k = -J_k^\top R_k$$

- Si $\|p_k\| < \tau$, hacer $res = 1$ y terminar el ciclo
- Hacer $z_{k+1} = z_k + p_k$.

3. Devolver $z_k, R_k, k, \|p_k\|$ y res.

---

1. Escriba una función que implementa el algoritmo anterior usando arreglos de Numpy.
2. Leer el archivo **puntos2D_1.npy** que contiene una matriz con dos columnas. La primer columna tiene los valores $x_1, x_2, ..., x_m$ y en la segunda columna los valores $y_1, y_2, ..., y_m$, de modo que cada par (x_i, y_i) es un dato. Queremos ajustar al conjunto de puntos (x_i, y_i) el modelo

   $$A \sin(wx + \phi)$$

   por lo que la función $R(\mathbf{z}) = R(A, w, \phi)$ está formada por los residuales

   $$r_i(z) = r_i(A, w, \phi) = A \sin(wx_i + \phi) - y_i$$

   para $i = 1, 2, ..., m$.

   Programe la función $R(\mathbf{z})$ con $\mathbf{z} = (A, w, \phi)$ y su Jacobiana $J(\mathbf{z})$.

   Nota: Puede programar estas funciones de la forma `funcion(z, paramf)`, donde `paramf` corresponda a la matriz que tiene los puntos (x_i, y_i). También puede pasar el arreglo `paramf` como arumento del algoritmo para que pueda evaluar las funciones.

3. Use el algoritmo con estas funciones $R(\mathbf{z})$ y $J(\mathbf{z})$, el punto inicial $\mathbf{z}_0 = (15, 0.6, 0)$ (esto es $A_0 = 15$, $w_0 = 0.6$ y $\phi_0 = 0$), un número máximo de iteraciones $N = 5000$ y una tolerancia $\tau = \sqrt{\epsilon_m}$ donde ϵ_m es el épsilon máquina.

- Imprima el valor inicial $f(\mathbf{z}_0) = \frac{1}{2} R(\mathbf{z}_0)^\top R(\mathbf{z}_0)$.
- Ejecute el algoritmo e imprima un mensaje que indique si el algoritmo converge dependiendo de la variable res.
- Imprima $\mathbf{z}_k$, $f(\mathbf{z}_k) = \frac{1}{2} R(\mathbf{z}_k)^\top R(\mathbf{z}_k)$, la norma $\|p_k\|$, y el número de iteraciones k realizadas.

4. Genere una gráfica que muestre a los puntos (x_i, y_i) y la gráfica del modelo $z_k[0]\sin(z_k[1]x + z_k[2])$, evaluando esta función en el intervalo

$$x \in [\min x_i, \max x_i]$$

5. De la gr'afica de los datos, e interpretando el parámetro A como la amplitud de la onda, se ve que $A_0 = 15$ es una buena inicialización para este paramétro. Para los otros parámetros también debe se debería usar su interpretación para dar buenos valores iniciales. Repita las pruebas con los puntos iniciales $\mathbf{z}_0 = (15, 1, 0)$ y $\mathbf{z}_0 = (15, 0.6, 1.6)$.

1.2.1 Solución:

[ ]:
```
# En esta celda puede poner el código de las funciones
# o poner la instrucción para importarlas de un archivo .py
```

[1]:
```
# Pruebas del algoritmo
```

—

1.3 Ejercicio 2 (5 puntos)

Programar el método de Levenberg-Marquart para mínimos cuadrados.

Dar la función de residuales $R(z)$, la función Jacobiana $J(z)$, un punto inicial z_0, un número máximo de iteraciones N, $\mu_{ref} > 0$ y la tolerancia $\tau > 0$.

1. Hacer $res = 0$ y construir la matriz identidad I de tamaño igual a la dimensión de z_0.
2. Calcular $R_0 = R(z_0)$
3. Calcular $J_0 = J(z_0)$
4. Calcular $f_0 = 0.5R_0^\top R_0$
5. Calcular $\mathbf{A} = J_0^\top J_0$ y $\mathbf{g} = J_0^\top R_0$
6. Calcular $\mu = \min\{\mu_{ref}, \max a_{ii}\}$, donde a_{ii} son los elementos de la diagonal de la matriz $\mathbf{A}$.
7. Para $k = 0, 1, ..., N$:

- Calcular $\mathbf{p}_k$ resolviendo el sistema

$$(\mathbf{A} + \mu\mathbf{I})\mathbf{p}_k = -\mathbf{g}$$

- Si $\|\mathbf{p}_k\| < \tau$, hacer $res = 1$ y terminar el ciclo.
- Calcular $\mathbf{z}_{k+1} = \mathbf{z}_k + \mathbf{p}_k$
- Calcular $\mathbf{R}_{k+1} = \mathbf{R}(\mathbf{z}_{k+1})$

- Calcular $f_{k+1} = 0.5\mathbf{R}_{k+1}^\top \mathbf{R}_{k+1}$
- Calcular el parámetro ρ (ver las notas de la clase 16)

$$\rho = (f_k - f_{k+1})/(q_k(\mathbf{x}_k) - q_k(\mathbf{x}_{k+1})) = (f_k - f_{k+1})/(-\mathbf{p}_k^\top \mathbf{g} + 0.5\mu_k \mathbf{p}_k^\top \mathbf{p}_k)$$

- Si $\rho < 0.25$, hacer $\mu = 2\mu$.
- Si $\rho > 0.75$, hacer $\mu = \mu/3$.
- Calcular $\mathbf{J}_{k+1} = \mathbf{J}(\mathbf{z}_{k+1})$
- Calcular $\mathbf{A} = \mathbf{J}_{k+1}^\top \mathbf{J}_{k+1}$ y $\mathbf{g} = \mathbf{J}_{k+1}^\top \mathbf{R}_{k+1}$.

8. Devolver el punto $\mathbf{z}_k$, f_k, k y res.

---

1. Escriba una función que implementa el algoritmo anterior usando arreglos de Numpy.
2. Aplique este algoritmo para resolver el problema del Ejercicio 1, imprimiendo la misma información y generando la gráfica correspondiente, usando $\tau = \sqrt{\epsilon_m}, N = 5000, \mu_{ref} = 0.001$ y los tres puntos iniciales

$$\mathbf{z}_0 = (15, 0.6, 0)$$
$$\mathbf{z}_0 = (15, 1.0, 0)$$
$$\mathbf{z}_0 = (15, 0.6, 1.6)$$

1.3.1 Solución:

[ ]:
```python
# En esta celda puede poner el código de las funciones
# o poner la instrucción para importarlas de un archivo .py
```

[ ]:
```python
# Pruebas realizadas a la función de residuales del Ejercicio 1
```

[ ]:
```python

```

—
