Métodos Numéricos II 2021

Lista 05

11.septiembre.2021

1. Implementar los siguientes métodos:

   - descenso gradiente con búsqueda en línea (Backtracking),
   - descenso gradiente con dirección de Newton,
   - descenso gradiente con dirección de Newton, con Hessiano aproximado por la matriz positiva definida más cercana B_k.

   En cada uno de los métodos, su función debe recibir los siguientes argumentos: la función objetivo f, el gradiente de la función objetivo df, el hessiano d^2f (de ser necesario), un punto inicial $\mathbf{x}_0 \in \mathbb{R}^n$, el tamaño de paso $\alpha_0 > 0$ inicial del Backtracking, el número máximo de iteraciones $maxIter$, la tolerancia ε, así como un criterio de paro.

   Para la tolerancia y el criterio de paro, mi sugerencia es que el usuario pueda ingresar el tipo de criterio de paro que desea (error absoluto en x: $||\mathbf{x}_k - \mathbf{x}_{k-1}||$; error absoluto en y: $|f(\mathbf{x}_k) - f(\mathbf{x}_{k-1})|$; error relativo, norma del gradiente $||\nabla f(\mathbf{x}_k)||$). Similarmente, el usuario debe poder indicar si desea que el Backtracking se detenga con las condición de Armijo, de Wolfe o de Goldstein.

   Como resultado, sus algoritmos deben devolver: la mejor solución encontrada $best$ $\mathbf{x}$ (la última de las aproximaciones calculadas); la secuencia de iteraciones $\mathbf{x}_k$; la secuencia de valores $f(\mathbf{x}_k)$; la secuencia de errores en cada paso (según el error de su criterio de paro).
   Además, es deseable indicar el número de iteraciones efectuadas por el algoritmo, y si se obtuvo o no convergencia del método, así como otras información que usted considere relevante.

2. Testar sus algoritmos del Ejercicio 1 con las siguientes funciones:

   a) La función de Wood $f : \mathbb{R}^4 \to \mathbb{R}$, dada por

   $$f(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2,$$

   con $\mathbf{x}_0 = (-3, 1, -3, 1)^T$, $\mathbf{x}^* = (1, 1, 1, 1)^T$, $f(\mathbf{x}^*) = 0$.

   b) La función de Rosembrock 100-dimensional $f : \mathbb{R}^{100} \to \mathbb{R}$, dada por

   $$f(\mathbf{x}) = \sum_{i=1}^{99} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right],$$

   con $\mathbf{x}_0 = (-1.2, 1, 1, \ldots, 1, -1.2, 1)^T$, $\mathbf{x}^* = (1, 1, \ldots, 1)^T$, $f(\mathbf{x}^*) = 0$.

   En cada uno de los casos, elaborar una tabla que compare los tres algoritmos implementados (con los mismo parámetros). En la tabla muestre un resumen del número de iteraciones requeridas para alcanzar convergencia, la solución aproximada $\mathbf{x}_k$ devuelta (sólo en (a)), el valor de $f(\mathbf{x}_k)$, el error de la solución. Muestre sólo las primeras 3 y últimas 3 iteraciones de cada método.

   Para cada ejemplo, elabore gráficas de error de aproximación (error vs. número de iteraciones), para comparar visualmente el desempeño de cada método.

3. **Aproximación por mínimos cuadrados con regularización de Tychonoff.**
Suponga que se cuenta con un conjunto de datos $\{(t_i, y_i)\}_{i=1}^n$ correspondientes a una serie de tiempo. Dichos datos se proporcionan en el archivo **yk.txt**. El primer valor $n = 128$ en el archivo indica el número de datos totales. Puede usar como intervalo de tiempo el rango $t_i \in (0, 128)$.

Aplique descenso gradiente para obtener el menor error de $f(\mathbf{x})$ para un problema de aproximación de datos (t_i, y_i), por mínimos cuadrados regularizados. Aquí, se asume que el modelo tiene un parámetro de regularización $\lambda > 0$, el cual controla el grado de suavizamiento de la función de aproximación.

Elabore una gráfica de (t_i, y_i) y $(t_i; x_i^*(\lambda))$ en la misma figura. Aquí $y_i = f(t_i)$ denota la función exacta dada por los datos, mientras que $x_i^* = f^*(t_i)$ denota la aproximación de mínimos cuadrados. La función a optimizar es

$$E(\mathbf{x}) = \sum_{i=1}^{n}(x_i - y_i)^2 + \lambda \sum_{i=1}^{n-1}(x_{i+1} - x_i)^2,$$

la cual corresponde al error cuadrático regularizado

$$||f(\mathbf{t}) - f^*(\mathbf{t})||_2^2 + \lambda\, ||\Delta f^*(\mathbf{t})||_2^2.$$

Considere los casos $\lambda \in \{0, 1, 10, 100\}$.

4. **Un problema de clasificación.**

a) Maximice la función de log-verosimilitud correspondiente a un problema de clasificación logística binaria de imágenes 0 y 1 en el conjunto MNIST.

$$\ell(\beta, \beta_0) = \sum_{i=1}^{n} \big[y_i\, \log \pi_i + (1 - y_i)\, \log(1 - \pi_i)\big],$$

donde $\pi_i = \pi(\beta, \beta_0, \mathbf{x}_i) = \dfrac{1}{1 + \exp(-\mathbf{x}_i^T \beta - \beta_0)}$.

Aquí, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ es el conjunto de datos, con $\mathbf{x}_i \in \mathbb{R}^{784}$, y $y_i \in \{0, 1\}$, se obtienen del conjunto de entrenamiento `train_set` (ver detalles abajo), sólo aquellos elementos con etiqueta $y_i \in \{0, 1\}$.

b) Usando $\widehat{\beta}, \widehat{\beta}_0$ hallados en (a), calcule el error de clasificación (en prueba)

$$E(\beta, \beta_0) = \frac{1}{n} \sum_{i=1}^{n} \Big|\mathbf{1}\big[\pi(\widehat{\beta}, \widehat{\beta}_0, \mathbf{x}_i) > 0.5\big] - y_i\Big|,$$

obtenido a partir del conjunto de prueba `test_set`, sólo con aquellos elementos con etiqueta $y_i \in \{0, 1\}$.

c) Para cada dígito 0 y 1, ilustre muestre 3 ejemplos de datos bien clasificados y 3 ejemplos de datos mal clasificados.

Los conjuntos de entrenamiento y prueba `train_set` y `test_set` se obtienen a partir del archivo `mnist.pkl.gz`. Este dataset puede leerse a partir del código

```
import cPickle, gzip, numpy
f = gzip.open('mnist.pkl.gz', 'rb')
train_set, valid_set, test_set = cPickle.load(f)
f.close()
```

`train_set[0]`, `train_set[1]`, `test_set[0]`, `test_set[1]` son arreglos numpy de (50000, 784), (50000,), (10000, 784), (10000,), resp.

También se puede usar cualquier otra forma de llamar a la base de datos en Python. Por ejemplo, la librería tensorflow incluye ya una copia de MNIST:

```
from tensorflow.keras.datasets import mnist
(train_X, train_y), (test_X, test_y) = mnist.load_data()
```