optim_tarea07

April 15, 2024

1 Curso de Optimización I

1.1 Tarea 7

Descripción:	Fechas
Fecha de publicación del documento:	**Abril 15, 2024**
Fecha límite de entrega de la tarea:	**Abril 23, 2024**

1.1.1 Indicaciones

- Envie el notebook con 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 (4 puntos)

Programar el método de Newton truncado descrito en el Algoritmo 1 y 2 de la Clase 20.

1. Programar la función que implementa el Algoritmo 1, que calcula una aproximación de la solución del sistema de Newton.

- Haga que la función devuelva la dirección $\mathbf{p}_k$ y el número de iteraciones realizadas.

2. Programar la función que implementa el Algoritmo 2.

- Use el algoritmo de backtracking con la condición de descenso suficiente para calcular el tamaño de paso α_k.
- Defina la variable binaria res de modo que $True$ si se cumple la condición de salida $\|\mathbf{g}_k\| < \tau$ y $False$ si termina por iteraciones.
- Calcule el promedio de las iteraciones realizadas por el Algoritmo 1
- Haga que la función devuelva $\mathbf{x}_k, \mathbf{g}_k, k, res$ y el promedio de la iteraciones realizadas por el Algoritmo 1.

3. Pruebe el algoritmo para minimizar las siguientes funciones usando los parámetros $N = 5000$, $\tau = \sqrt{n}\epsilon_m^{1/3}$, donde n es la dimensión de la variable $\mathbf{x}$ y ϵ_m es el épsilon máquina. Para backtracking use $\rho = 0.5$, $c_1 = 0.001$ y el número máximo de iteraciones $N_b = 500$.

   En cada caso imprima los siguientes datos:

- la dimensión n,
- $f(\mathbf{x}_0)$,
- el número k de iteraciones realizadas,
- $f(\mathbf{x}_k)$,
- las primeras y últimas 4 entradas del punto $\mathbf{x}_k$ que devuelve el algoritmo,
- la norma del vector gradiente $\mathbf{g}_k$,
- el promedio del número de iteraciones realizadas por el Algoritmo 1.
- la variable *res* para saber si el algoritmo puedo converger.

Función de cuadrática 1: Para $\mathbf{x} = (x_1, x_2, ..., x_n)$

- $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}_1 \mathbf{x} - \mathbf{b}_1^\top \mathbf{x}$, donde $\mathbf{A}_1$ y $\mathbf{b}_1$ están definidas por

$$\mathbf{A}_1 = n\mathbf{I} + \mathbf{1} = \begin{bmatrix} n & 0 & \cdots & 0 \\ 0 & n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & n \end{bmatrix} + \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}, \qquad \mathbf{b}_1 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix},$$

donde $\mathbf{I}$ es la matriz identidad y $\mathbf{1}$ es la matriz llena de 1's, ambas de tamaño n, usando los puntos iniciales
- $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{10}$ - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{100}$ - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{1000}$

Función de cuadrática 2: Para $\mathbf{x} = (x_1, x_2, ..., x_n)$

- $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}_2 \mathbf{x} - \mathbf{b}_2^\top \mathbf{x}$, donde $\mathbf{A}_2 = [a_{ij}]$ y $\mathbf{b}_2$ están definidas por

$$a_{ij} = exp\left(-0.25(i-j)^2\right), \qquad \mathbf{b}_2 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

usando los puntos iniciales: - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{10}$ - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{100}$ - $\mathbf{x}_0 = (0, ..., 0) \in \mathbb{R}^{1000}$

Función de Beale : Para $\mathbf{x} = (x_1, x_2)$

$$f(\mathbf{x}) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2.$$

- $\mathbf{x}_0 = (2, 3)$

Función de Himmelblau: Para $\mathbf{x} = (x_1, x_2)$

$$f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$

- $\mathbf{x}_0 = (2, 4)$

Función de Rosenbrock: Para $\mathbf{x} = (x_1, x_2, ..., x_n)$

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \quad n \geq 2.$$

- $\mathbf{x}_0 = (-1.2, 1.0) \in \mathbb{R}^2$
- $\mathbf{x}_0 = (-1.2, 1.0, ..., -1.2, 1.0) \in \mathbb{R}^{20}$
- $\mathbf{x}_0 = (-1.2, 1.0, ..., -1.2, 1.0) \in \mathbb{R}^{40}$

1.2.1 Solución:

[ ]:

[ ]:

1.3 Ejercicio 2 (3 puntos)

Programar las funciones que calcule el gradiente y la Hessiana usando el método de diferencias finitas.

1. Programe la función que calcule una aproximación del gradiente de una función $f(\mathbf{x})$ en un punto $\mathbf{x} \in \mathbb{R}^n$ dado usando el esquema de diferencias finitas hacia adelante (Página 20 de la Clase 20).

- La función recibe como parámetros la función f, el punto $\mathbf{x}$ y el incremento h y devuelve el arreglo de tamaño n con las aproximaciones de aproximaciones de las derivadas parciales en el punto $\mathbf{x}$.

2. Programe la función que calcule una aproximación de la Hessiana de una función $f(\mathbf{x})$ en un punto $\mathbf{x} \in \mathbb{R}^n$ dado usando el esquema de diferencias finitas de la Página 22 de la Clase 20.

- La función recibe como parámetros la función f, el punto $\mathbf{x}$ y el incremento h y devuelve una matriz simétrica de tamaño n que tiene las aproximaciones de las segundas derivadas parciales de f en el punto $\mathbf{x}$.

3. Modifique la función `errorRelativo_grad` para reportar estadísticas del error relativo de la implementación del gradiente analítico `gradf` de una función respecto al gradiente calculado con `autograd`, para que mida el error relativo entre la función `gradf` y la aproximación del gradiente usando diferencias finitas. Hay que agregar como parámetro de `errorRelativo_grad` el incremento h para que se pueda llamar la función del Punto 1.

4. Programar la función `errorRelativo_hess`, similar a la función del punto anterior, para que reporte estadísticas del error relativo entre una función que calcula la Hessiana de f de manera analítica en un punto $\mathbf{x}$ y la aproximación de la Hessiana en $\mathbf{x}$ usando diferencias finitas.

5. Pruebe las funciones `errorRelativo_grad` con cada una de las funciones del Ejercicio 1 usando $h = 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}$. ¿Cuál es el valor de h que conviene usar para aproximar el gradiente y cuál para aproximar la Hessiana?

1.3.1 Solución:

[ ]:
```
def errorRelativo_grad(fncf, gradf, n, nt):
    '''
    Imprime estadísticas sobre el error relativo entre el gradiente analítico
    de una función y el cálculado con autograd.
    fncf  - Función f
    gradf - Implementación del gradiente analítico de la función f
    n     - Dimensión de la variable n
    nt    - Número de puntos aleatorios en donde se comparan los gradientes
    '''
    ve = np.zeros(nt)
    print('\nErrores relativos en el cálculo del gradiente:')
    g_f = egrad(fncf)  # Funcion gradiente generada con autograd
    for it in range(nt):
        x0  = np.random.randn(n)
        g0  = gradf(x0)
        ga  = g_f(x0)
        ve[it] = np.linalg.norm(g0-ga)/np.linalg.norm(ga)
    print('Min: %.2e   Media: %.2e    Max: %.2e' %(np.min(ve), np.mean(ve), np.
↪max(ve)))
```

[ ]:
```

```

1.4 Ejercicio 3 (3 puntos)

Seleccionar un artículo para el proyecto final.

- El proyecto final se puede presentar de manera individual o en equipo formado por dos estudiantes.
- La entrega del proyecto consiste programar el algoritmo descrito en el artículo seleccionado y realizar pruebas para reproducir algunos resultados presentados en el artículo o diseñar los experimentos de prueba. El objetivo es mostrar las ventajas o limitaciones que tiene el algoritmo propuesto.
- Es válido delimitar el alcance, de manera que si aparecen varios algoritmos en el artículo, se puede seleccionar alguno de ellos para su implementación y validación.
- Hay que elaborar un reporte en el que se dé una introducción, algunos fundamentos teóricos, el planteamiento del problema, la descripción del algoritmo, los resultados obtenidos y las conclusiones.
- Hay que hacer una presentación de unos 15 minutos en el día acordado y entregar el reporte, el código y las pruebas realizadas.

- Se puede entregar un notebook como el reporte y usarlo en la presentación, para que no tener que elaborar un documento con el reporte, otro con el script del código y pruebas y otro para la presentación.
- Habrá dos fechas de entrega. La primera fecha es para los estudiantes de posgrado que será entre el 27 de mayo y el 4 de junio. La segunda fecha es para los estudiantes de licenciatura que será entre el 3 de junio y el 10 de junio.
- Si el equipo está formado por un estudiante de licenciatura y otro de posgrado tendrá que presentar el proyecto en la primera fecha.
- Para la selección se puede tomar uno de los artículos de la lista que se presenta a continuación.
- Estos artículos son una referencia. También pueden proponer algún artículo adicional, pero recomienda que cuiden que para entenderlo no tengan que revisar otras fuentes o que tengan que implementar algoritmos que requieran de temas que no fueron cubiertos en el curso y que les consuma demasiado tiempo hacer esa revisión, por ejemplo, en temas de optimización combinatoria, entera, mixta, multiobjetivo, etc.

1. Escriba el nombre de los miembros del equipo junto con el nombre del programa académico.
2. Escriba el título del artículo seleccionado
3. Si no es un artículo de la lista o que esté en el Classroom, agregue el PDF como parte de la entrega de la Tarea 7.

1.4.1 Lista de artículos

1. An improvement of the Goldstein line search. Arnold NeumaierMorteza Kimiaei. 2023. https://optimization-online.org/2022/11/an-efficient-gradient-free-line-search/
2. A harmonic framework for stepsize selection in gradient methods. Giulia FerrandiMichiel E. HochstenbachNataša Krejić. 2022. https://optimization-online.org/2022/02/8803/
3. An Adaptive Trust-Region Method Without Function Evaluations. Geovani Nunes GrapigliaGabriel Stella. 2022. https://optimization-online.org/2022/02/8787/
4. Accelerated Scaled Memory-less SR1 method for Unconstrained Optimization. Neculai Andrei. 2021. https://camo.ici.ro/neculai/XXITR5.pdf
5. Secant Penalized BFGS: A Noise Robust Quasi-Newton Method Via Penalizing The Secant Condition. Brian Irwin, Eldad Haber. https://arxiv.org/abs/2010.01275
6. Regularized Step Directions in Nonlinear Conjugate Gradient Methods. Cassidy K. Buhler, Hande Y. Benson, David F. Shanno. 2023. https://arxiv.org/abs/2110.06308
7. A Levenberg-Marquardt Method for Nonsmooth Regularized Least Squares. Aleksandr Y. Aravkin, Robert Baraldi, Dominique Orban. 2023. https://arxiv.org/abs/2301.02347
8. Globally linearly convergent nonlinear conjugate gradients without Wolfe line search. Arnold NeumaierMorteza KimiaeiBehzad Azmi. 2023. https://optimization-online.org/2022/12/globally-linearly-convergent-nonlinear-conjugate-gradients-without-wolfe-line-search/
9. A nonlinear conjugate gradient method with complexity guarantees and its application to nonconvex regression. Rémi Chan–Renous-Legoubin, Clément W. Royer. 2022. https://arxiv.org/abs/2201.08568
10. Iteration Complexity of Fixed-Step Methods by Nesterov and Polyak for Convex Quadratic Functions. Melinda Hagedorn, Florian Jarre. 2022. https://arxiv.org/abs/2211.10234
11. Subsampled cubic regularization method for finite-sum minimization. Max L. N. Gonçalves. 2022. https://files.cercomp.ufg.br/weby/up/922/o/Inexact_CNMSubsampled16November2022.pdf

12. Optimized convergence of stochastic gradient descent by weighted averaging. Melinda Hagedorn, Florian Jarre. 2022. https://arxiv.org/abs/2209.14092
13. Two efficient gradient methods with approximately optimal stepsizes based on regularization models for unconstrained optimization. Zexian Liu, Wangli Chu, Hongwei Liu. https://arxiv.org/abs/1907.01794
14. A Trust Region Method for the Optimization of Noisy Functions. Shigeng Sun, Jorge Nocedal. 2022. https://arxiv.org/abs/2201.00973
15. A modified quasi-Newton method for nonlinear equations. Xiaowei Fang, Qin Ni, Meilan Zeng. 2018. https://drive.google.com/file/d/13s8ey-LVioDjx3BFksWTUzccRXV9sEbG/view?usp=sharing
16. A minibatch stochastic Quasi-Newton method adapted for nonconvex deep learning problems. Joshua D. GriffinMajid JahaniMartin TakacSeyedalireza YektamaramWenwen Zhou. 2022. https://optimization-online.org/2022/01/8760/
17. Nonlinear conjugate gradient for smooth convex functions. Sahar Karimi, Stephen Vavasis. 2024. https://arxiv.org/abs/2111.11613
18. Quadratic Regularization Methods with Finite-Difference Gradient Approximations. Geovani Nunes Grapiglia. 2021. https://optimization-online.org/wp-content/uploads/2021/11/8665.pdf
19. Adaptive Finite-Difference Interval Estimation for Noisy Derivative-Free Optimization. Hao-Jun Michael Shi, Yuchen Xie, Melody Qiming Xuan, Jorge Nocedal. 2021. https://arxiv.org/abs/2110.06380
20. Full-low evaluation methods for derivative-free optimization. Albert S. Berahas, Luis Nunes, Vicente Oumaima Sohab. 2021. https://arxiv.org/abs/2107.11908
21. A stochastic first-order trust-region method with inexact restoration for finite-sum minimization. Stefania Bellavia, Natasa Krejic, Benedetta Morini, Simone Rebegoldi. 2022. https://arxiv.org/abs/2107.03129
22. Robust Conjugate Gradient Methods for Non-smooth Convex Optimization and Image Processing Problems. Salar Farahmand-Tabar, Fahimeh Abdollahi, and Masoud Fatemi. https://drive.google.com/file/d/1ItSZ_7N0QKJLvqI8jhyAKU532oxWMmwg/view?usp=sharing
23. A family of hybrid conjugate gradient method with restart procedure for unconstrained optimizations and image restorations. Xianzhen Jiang, Xiaomin Ye, Zefeng Huang, Meixing Liu. 2023 https://drive.google.com/file/d/1h1TQpydDxkYBF0G_jd7O7-9J69X-V-dI/view?usp=sharing
24. A modified Broyden-like quasi-Newton method for nonlinear equations. Weijun Zhou, Li Zhang. 2020. https://drive.google.com/file/d/152StQd3SVdNUXRSSboABFpjHbqEVxUP9/view?usp=sharing

1.4.2 Respuesta:

[ ]: