Tarea 4 Optimización

Roberto Vásquez Martínez
Profesor: Joaquín Peña Acevedo

06/Marzo/2022

1 Ejercicio 1 (5 puntos)

Programar el método de descenso máximo con tamaño de paso exacto para minimizar funciones cuadráticas:

$$f(x) = \frac{1}{2}x^\top \mathbf{A}x - b^\top x,$$

donde $\mathbf{A} \in \mathbb{R}^{n\times n}$ y $x \in \mathbb{R}^n$.

Dado el vector b, la matriz $\mathbf{A}$, un punto inicial x_0, un número máximo de iteraciones N, la tolerancia $\tau > 0$. Fijar $k = 0$ y repetir los siguientes pasos:

1. Calcular el gradiente en el punto x_k,

$$g_k = \nabla f(x_k) = \mathbf{A}x_k - b$$

2. Si $\|g_k\| < \tau$, entonces x_k es (casi) un punto estacionario. Hacer $res = 1$ y terminar el ciclo.
3. Elegir la dirección de descenso como $p_k = -g_k$.
4. Calcular el tamaño de paso α_k que minimiza el valor de la función

$$\phi_k(\alpha) = f(x_k + \alpha p_k)$$

   es decir, calcular

$$\alpha_k = -\frac{g_k^\top p_k}{p_k^\top \mathbf{A} p_k}$$

5. Calcular el siguiente punto de la secuencia como

$$x_{k+1} = x_k + \alpha_k p_k$$

6. Si $k + 1 \geq N$, hacer $res = 0$ y terminar.
7. Si no, hacer $k = k + 1$ y volver el paso 1.
8. Devolver el punto x_k, $f_k = \frac{1}{2}x_k^\top \mathbf{A}x_k - b^\top x_k$, g_k, k y res.

---

1. Escriba una función que implementa el algoritmo anterior usando arreglos de Numpy.

2. Escriba una función para probar el funcionamiento del método de descenso máximo. Esta función debe recibir como parámetros el nombre de un archivo `.npy` que contiene las entradas de una matriz cuadrada $\mathbf{A}$, el vector b, un punto inicial x_0, el número máximo de iteraciones N y la tolerancia τ.

- Esta función debe crear la matriz $\mathbf{A}$ y el vector b leyendo los archivos de datos.
- Obtener el número de filas r de la matriz e imprimir este valor.
- Compruebe que la matriz es simétrica y definida positiva calculando e imprimiendo el valor $\|\mathbf{A} - \mathbf{A}^\top\|$ y su eigenvalor más pequeño (use la función `numpy.linalg.eig()`).
- Ejecutar la función del Inciso 1.
- Dependiendo del valor de la variable *res*, imprima un mensaje que diga que el algoritmo convergió ($res = 1$) o no ($res = 0$).
- Imprimir k, f_k, la norma de g_k, y los primeros 3 y últimos 3 elementos del arreglo x_k.
- Calcule directamente el minimizador resolviendo la ecuación $Ax_* = b$ e imprima el valor del error $\|x_k - x_*\|$.

3. Pruebe la función del Inciso 2 usando $N = 1000$, la tolerancia $\tau = \epsilon_m^{1/3}$, donde ϵ_m es el épsilon de la máquina, y los arreglos que se incluyen en el archivo datosTarea04.zip, de la siguiente manera:

Matriz	Vector	Punto inicial
A1.npy	b1.npy	$x_0 = (0, -5)$
A1.npy	b1.npy	$x_0 = (7045, 7095)$
A2.npy	b2.npy	$x_0 = (0, 0, \ldots, 0) \in \mathbb{R}^{500}$
A2.npy	b2.npy	$x_0 = (10000, 10000, \ldots, 10000) \in \mathbb{R}^{500}$

1.1 Solución:

En la siguiente celda declaramos el PATH de los archivos `.npy` que usaremos para probar el método de descenso máximo con paso exacto para la funciones cuadráticas de la forma

$$f(x) = \frac{1}{2} x^\top \mathbf{A} x - b^\top x,$$

donde $A \in \mathbb{R}^{n \times n}$ es una matriz simétrica positiva definida.

Importamos el módulo `lib_t4` donde se encuentran las funciones `grad_max_quadratic` y `proof_grad_max_quadratic`, que implementan los numerales 1 y 2, respectvamente.

[1]:
```
# Path de los archivos con los datos
data_A1='/datosTarea04/A1.npy'
data_A2='/datosTarea04/A2.npy'
data_b1='/datosTarea04/b1.npy'
data_b2='/datosTarea04/b2.npy'
```

1.1.1 Función 1

A continuación realizamos las pruebas correspondiente a dos diferentes condiciones iniciales x_0 tomando los datos de la matriz en `A1.npy` y el vector `b1.npy` para definir la función cuadrática. En

cada una de las pruebas se utiliza como tolerancia $\tau = \epsilon_m^{1/3}$ y un número máximo de iteraciones $N = 1000$, donde ϵ_m es el épsilon de la máquina.

Para $x_0 = (0, -5)$ tenemos el siguiente resultado

[2]:
```
import lib_t4
import importlib
importlib.reload(lib_t4)
from lib_t4 import *
x0=np.array([0.0,-5.0])
tol=np.finfo(float).eps**(1/3)
N=1000
proof_grad_max_quadratic(data_A1,data_b1,x0,N,tol)
```

```
El número de filas de la matriz A es 2
||A-A.T||=  0.0
Es una matriz simétrica
El valor propio más pequeño es:  0.10000000000000003
La matriz A es positiva definida
El método de descenso máximo con paso exacto CONVERGE
k =  69
fk =  -62.749999999933024
||gk|| =  4.767340294300877e-06
xk =  [-24.49997287  25.49997743]
||xk-x*|| =  3.528998540703229e-05
```

Para la condición inicial $x_0 = (7045, 7095)$ obtenemos lo siguiente

[3]:
```
x0=np.array([7045.0,7095.0])
proof_grad_max_quadratic(data_A1,data_b1,x0,N,tol)
```

```
El número de filas de la matriz A es 2
||A-A.T||=  0.0
Es una matriz simétrica
El valor propio más pequeño es:  0.10000000000000003
La matriz A es positiva definida
El método de descenso máximo con paso exacto CONVERGE
k =  1
fk =  -62.75
||gk|| =  6.425429159208664e-13
xk =  [-24.5  25.5]
||xk-x*|| =  9.131096203816022e-13
```

Aquí llama la atención como en le primer paso se llegó al mínimo de la función, incluso con mayor exactitud que con la primer condición inicial como se puede constantar en el valor de $\|x_k - x_*\|$ para cada caso.

1.1.2 Función 2

Ahora usaremos los datos de la matriz en `A2.npy` y el vector `b2.npy` para definir la función cuadrática.

Ejecutando el algoritmo de descenso máximo con paso exacto para dicha función con la codición inicial $x_0 = (0, 0, \ldots, 0) \in \mathbb{R}^{500}$ obtenemos lo siguiente

[4]:
```
x0=np.zeros(500)
proof_grad_max_quadratic(data_A2,data_b2,x0,N,tol)
```

```
El número de filas de la matriz A es 500
||A-A.T||=  1.5063918215255853e-14
Es una matriz simétrica
El valor propio más pequeño es:  0.09999999999999767
La matriz A es positiva definida
El método de descenso máximo con paso exacto CONVERGE
k =  332
fk =  -5239.541412076964
||gk|| =  5.996601797613609e-06
Primeras 3 coordenadas de xk son:  [-11.61784712   5.44982336   0.96506345]
Últimas 3 coordenadas de xk son:  [ -1.8852386  -10.34321936  -4.15697837]
||xk-x*|| =  3.9297511569442596e-05
```

Con la condición inicial $x_0 = (10000, 10000, \ldots, 10000) \in \mathbb{R}^{500}$ obtenemos

[5]:
```
x0=np.full(500,10000.0)
proof_grad_max_quadratic(data_A2,data_b2,x0,N,tol)
```

```
El número de filas de la matriz A es 500
||A-A.T||=  1.5063918215255853e-14
Es una matriz simétrica
El valor propio más pequeño es:  0.09999999999999767
La matriz A es positiva definida
El método de descenso máximo con paso exacto CONVERGE
k =  453
fk =  -5239.541412076969
||gk|| =  5.700360427845806e-06
Primeras 3 coordenadas de xk son:  [-11.61784726   5.4498234    0.96506346]
Últimas 3 coordenadas de xk son:  [ -1.8852385  -10.34321955  -4.15697837]
||xk-x*|| =  3.859553615080582e-05
```

A diferencia de la primer función cuadrática no hay una diferencia tan radical. Con la segunda condición inicial se necesitan más iteraciones para alcanzar la convergencia que en la primero, pero al final los resultados son bastante parecidos incluso en la distancia con el punto estacionario x_*,

2 Ejercicio 2 (5 puntos)

Programar el método de descenso máximo con tamaño de paso seleccionado por la estrategia de backtracking:

Algoritmo de descenso máximo con backtracking:

Dada una función $f : \mathbb{R}^n \to \mathbb{R}$, su gradiente $g : \mathbb{R}^n \to \mathbb{R}^n$, un punto inicial x_0, un número máximo de iteraciones N, una tolerancia $\tau > 0$. Fijar $k = 0$ y repetir los siguientes pasos:

1. Calcular el gradiente en el punto x_k:

$$g_k = \nabla f(x_k) = g(x_k)$$

2. Si $\|g_k\| < \tau$, x_k es un aproximadamente un punto estacionario, por lo que hay que hacer $res = 1$ y terminar el ciclo.
3. Eligir la dirección de descenso como $p_k = -g_k$.
4. Calcular el tamaño de paso α_k mediante la estrategia de backtraking, usando el algoritmo que describe más adelante.
5. Calcular el siguiente punto de la secuencia como

$$x_{k+1} = x_k + \alpha_k p_k$$

6. Si $k + 1 > N$, hacer $res = 0$ y terminar.
7. Si no, hacer $k = k + 1$ y volver el paso 1.
8. Devolver el punto x_k, $f_k = f(x_k)$, g_k, k y res.

** Algoritmo de backtracking **

Backtracking(f, f_k, g_k, x_k, p_k, α_{ini}, ρ, c)

El algoritmo recibe la función f, el punto x_k, $f_k = f(x_k)$, la dirección de descenso p_k, un valor inicial α_{ini}, $\rho \in (0, 1)$, $c \in (0, 1)$.

Fijar $\alpha = \alpha_{ini}$ y repetir los siguientes pasos:

1. Si se cumple la condición
$$f(x_k + \alpha p_k) \leq f_k + c\alpha g_k^\top p_k,$$
terminar el ciclo devolviendo
2. Hacer $\alpha = \rho\alpha$ y regresar al paso anterior.

---

1. Escriba una función que implementa el algoritmo de backtracking.
2. Escriba la función que implementa el algoritmo de máximo descenso con búsqueda inexacta, usando backtraking. Tiene que recibir como paramétros todos los elementos que se listaron para ambos algoritmos.

3. Escriba una función para probar el funcionamiento del método de descenso máximo. Esta función debe recibir la función f, la función g que devuelve su gradiente, el punto inicial x_0, el número
máximo de iteraciones N, la tolerancia $\tau > 0$ y el factor ρ del algoritmo de backtracking.

- Fijar los parámetros $\alpha_{ini} = 2$ y $c = 0.0001$ del algoritmo de backtracking.
- Ejecutar la función del Inciso 2.
- Dependiendo del valor de la variable *res*, imprima un mensaje que diga que el algoritmo convergió ($res = 1$) o no ($res = 0$)
- Imprimir k, x_k, f_k y la norma de g_k.

4. Pruebe la función del Inciso 3 usando $N = 10000$, $\rho = 0.8$, la tolerancia $\tau = \epsilon_m^{1/3}$, donde ϵ_m es el épsilon de la máquina. Aplique esta función a:

- La función de Rosenbrock, descrita en la Tarea 3, usando como punto inicial $x_0 = (-1.2, 1)$ y $x_0 = (-12, 10)$. Como referencia, el minimizador de la función es $x_* = (1, 1)$.

5. Repita el inciso anterior con $\rho = 0.5$.

2.1 Solución:

De manera similar al ejercicio anterior, importamos el módulo `lib_t4` donde se encuentran las funciones `backtracking`, `grad_max` y `proof_grad_max` que son las implementaciones de lo que se solicita en los numerales 1,2 y 3, respectivamente.

Realizamos las pruebas del método de descenso máximo con tamaño de paso obtenido a través del *algoritmo backtracking*.

Fijamos un número máximo de iteraciones $N = 10,000$ y una tolerancia $\tau = \epsilon_m^{1/3}$.

2.1.1 $\rho = 0.8$

En primer lugar, probamos este método de optimización con la *Función de Rosenbrock* con la condición inicial $x_0 = (-1.2, 1)$ y tomando $\rho = 0.8$, el resultado es el siguiente

[6]:
```
importlib.reload(lib_t4)
from lib_t4 import *
tol=np.finfo(float).eps**(1/3)
N=10000
rho=0.8
x0=np.array([-1.2,1.0])
proof_grad_max(f_Rosenbrock,grad_Rosenbrock,x0,N,tol,rho)
```

```
El algoritmo de descenso máximo con backtracking NO CONVERGE
k =  10000
xk =  [1.00079208 1.00158391]
fk =  6.274640640496038e-07
||gk|| =  0.0019653292817284995
```

Con el mismo factor $\rho = 0.8$ pero con la condición inicial $x_0 = (-12, 10)$ obtenemos

[7]:
```
x0=np.array([-12.0,10.0])
proof_grad_max(f_Rosenbrock,grad_Rosenbrock,x0,N,tol,rho)
```

```
El algoritmo de descenso máximo con backtracking NO CONVERGE
k =  10000
xk =  [ 3.92075554 15.37404809]
fk =  8.531110164767732
||gk|| =  2.823173435454123
```

Observamos que en ambos casos se alcanza el máximo número de iteraciones. Con la condición inicial $x_0 = (-12, 10)$, al estar más lejos de el óptimo $x_* = (1, 1)$ el resultado final igual queda más lejos de x_* que con la primer condición inicial, pues con esta condición inicial el método de descenso máximo en las 10000 iteraciones queda mucho más cerca del valor x_*.

2.1.2 $\rho = 0.5$

Ahora cambiamos sólo el factor por el cual aumentamos la velocidad de reducción de el tamaño de paso en el *algoritmo backtracking* a $\rho = 0.5$.

Para la condición inicial $x_0 = (-1.2, 1)$ obtenemos el siguiente resultado

[8]:
```
rho=0.5
x0=np.array([-1.2,1.0])
proof_grad_max(f_Rosenbrock,grad_Rosenbrock,x0,N,tol,rho)
```

```
El algoritmo de descenso máximo con backtracking NO CONVERGE
k =  10000
xk =  [0.99998357 0.99996704]
fk =  2.7097756887567074e-10
||gk|| =  2.377226279771169e-05
```

Para la condición inicial $x_0 = (-12, 10)$ el resultado es

[9]:
```
x0=np.array([-12.0,10.0])
proof_grad_max(f_Rosenbrock,grad_Rosenbrock,x0,N,tol,rho)
```

```
El algoritmo de descenso máximo con backtracking NO CONVERGE
k =  10000
xk =  [2.85989511 8.18266786]
fk =  3.46055510931203
||gk|| =  1.121519863018428
```

Al igual que con el factor $\rho = 0.8$ con ambas condiciones se alcanza el máximo de iteraciones, sin embargo el valor final de la función de Rosenbrock en ambos casos queda mucho más cerca del valor óptimo que es $f(x_*) = 0$, lo que sugiere que quizás el factor $\rho = 0.8$ en el algoritmo backtracking deja aún muy largo el tamaño de paso que puede ocasionar que rebote la búsqueda en línea.

Finalmente, vemos que si aumentamos la velocidad a la que el tamaño de paso $\alpha \to 0$ tomando $\rho = 0.4$. Para la condición inicial $x_0 = (-1.2, 1)$ el resultado es

[10]:
```
rho=0.4
x0=np.array([-1.2,1.0])
proof_grad_max(f_Rosenbrock,grad_Rosenbrock,x0,N,tol,rho)
```

```
El algoritmo de descenso máximo con backtracking CONVERGE
k =  9989
xk =  [1.00000616 1.00001235]
fk =  3.80249328045783e-11
||gk|| =  5.974110497205854e-06
```

Para la condición inicial $x_0 = (-12, 10)$ los resultados son los siguientes

[11]:
```
x0=np.array([-12.0,10.0])
proof_grad_max(f_Rosenbrock,grad_Rosenbrock,x0,N,tol,rho)
```

```
El algoritmo de descenso máximo con backtracking NO CONVERGE
k =  10000
xk =  [1.93435739 3.74235441]
fk =  0.873061658777709
||gk|| =  0.6240981756328119
```

A diferencia de los otros factores ρ con $\rho = 0.4$ tenemos convergencia para la primer condición inicial. Análogamente, para la segunda condición inicial estamos más cerca del valor óptimo de la función de Rosenbrock que con los otros factores ρ.

Observando la gráfica de la función de Rosenbrock en la *Tarea 3*, podemos concluir que este comportamiento se puede deber a que en el rayo $\lambda(-1.2, 1.0)$ con $\lambda \in \mathbb{R}$ se encuentra un precipicio algo pronunciado, entonces para pasos grandes nos podemos salir de ese valle y alejarnos del óptimo de forma más fácil que con tamaños de paso más pequeños, por lo que eso explica que el comportamiento del método de descenso máximo mejore cuando se acelera la velocidad con la que el tamaño de paso $\alpha \to 0$ en el *algoritmo backtracking*.