OPTIMIZACIÓN II
Práctica 3

Francisco J. Aragón Artacho
Dpto. Matemáticas, Universidad de Alicante

17-18 de marzo de 2022

Por la Proposición 7, si p es una dirección de descenso en el punto x para la función f, la cual está acotada inferiormente sobre la semirrecta $\{x + \alpha p \mid \alpha > 0\}$, el siguiente algoritmo encuentra una longitud de paso que satisface las condiciones de Wolfe.

Algoritmo 1: Algoritmo de bisección para las condiciones de Wolfe

Elegir $0 < c_1 < c_2 < 1$. *Tomar* $a = 0$, $b = 2$, *continua* = *true*.
while $f(x + bp) \leq f(x) + c_1 b f'(x; p)$ **do**
  $b = 2b$
end
while *continua* **do**
  $\alpha = \frac{1}{2}(a + b)$
  if $f(x + \alpha p) > f(x) + c_1 \alpha f'(x; p)$ **then**
    $b = \alpha$
  else if $f'(x + \alpha p; p) < c_2 f'(x; p)$ **then**
    $a = \alpha$
  else
    continua = *false*
  end
end

En esta práctica aplicaremos el método del descenso más rápido para reducir el ruido de una imagen. Las imágenes digitalizadas están compuestas por los 3 colores de luz primarios (RGB: Red - rojo, Green - verde y Blue - azul). Así, una imagen de $m \times n$ píxeles estará compuesta por 3 matrices en $[0, 1]^{m \times n}$, donde cada elemento de la matriz representa la intensidad del color correspondiente.

Por consiguiente, trataremos con matrices en vez de vectores. Usaremos la norma matricial de Frobenius

$$\|A\|_F := \sqrt{\mathrm{tr}(\mathrm{A}^\mathrm{T}\mathrm{A})}$$

que está inducida por el producto interno de Frobenius

$$\langle A, B \rangle := \mathrm{tr}(\mathrm{A}^\mathrm{T}\mathrm{B}).$$

El espacio de las matrices $\mathbb{R}^{m \times n}$ dotado con la norma de Frobenius es un espacio de Hilbert. Si $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ es una función diferenciable y $x, p \in \mathbb{R}^{m \times n}$, entonces se prueba fácilmente que

$$f'(x; p) = \langle \nabla f(x), p \rangle. \tag{1}$$

Para reducir el ruido de la imagen, minimizaremos la función $f : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \times \mathbb{R}_+ \to \mathbb{R}$ *respecto a su primera variable, donde*

$$f(x, y, \lambda) = \frac{1}{2}\|x - y\|_F^2 + \lambda TV(x),$$

siendo y *la imagen con ruido,* x *cualquier imagen y* TV *la* norma de variación total *de* x*. El primer término de la función* f *mide la distancia de la imagen* x *a la imagen con ruido* y*, mientras que el segundo término mide el cambio entre los píxeles próximos de* x *(es decir, la "suavidad" de la imagen* x*).*

Carga el programa de Python `Pr3.py` y añade los comandos oportunos para:

1. Crear una función llamada `Wolfe` cuyos argumentos sean `f`, `g`, `x`, `p`, `c1` y `c2`. Define, por defecto, `c1=1e-4` y `c2=0.9`. El argumento `g` corresponde al gradiente de cualquier función `f`. Esta función debe basarse en el Algoritmo 1 de los apuntes (pág. 190 del libro) y debe devolver un tamaño de paso `alpha` que verifique las condiciones de Wolfe. Recuerda que debes utilizar la expresión (1) para calcular la derivada direccional.

2. Utilizando el comando `plt.imread`, carga la imagen con ruido gaussiano `foto_noisy.jpg` y guarda el resultado de dividir el *array* entre 255 en la variable `I`.

3. Con el objetivo de reducir el ruido de la imagen, programa 50 iteraciones del método del descenso más rápido aplicado a la función `f` predefinida dentro del programa. El gradiente de f respecto a x viene predefinido también en la función `Gradf` (al igual que f, es una función de tres variables x, y y λ). El tamaño de paso será elegido con la función `Wolfe` programada en el apartado 1. Se debe aplicar el método del descenso más rápido 3 veces (a cada una de las componentes RGB de la imagen por separado). Elige un parámetro $\lambda \in \,]0, 1[$ que reduzca el ruido de la imagen de manera satisfactoria (a mayor λ, mayor suavizado de la imagen). Toma como punto inicial cada una de las correspondientes componentes RGB de la imagen con ruido.

4. Junta las 3 componentes RGB de la imagen en un *array* tridimensional llamado `im` (créalo primero como un array de ceros con `np.zeros`). Muestra la imagen con el comando `plt.imshow` y guárdala en el archivo `foto_restaurada.jpg` usando `plt.imsave`.

5. Utiliza el comando `plt.subplot` para mostrar en una misma figura dos gráficas: en la de arriba mostraremos los valores de `f` para cada componente RGB en cada iteración y en la de abajo representaremos el valor de la norma de Frobenius de `Gradf` para cada componente RGB. Debe quedar una figura similar a la siguiente: