Práctica 1 (PARTE I): Minimización de funciones multivariantes no restringidas con Matlab.

Fechas

Inicio: 3/03/05
Entrega: no hay entrega de esta parte

Objetivo: Uso de las funciones fminsearch y fminunc

fminsearch

El algoritmo implementado en esta función es el método Simplex (Nelder y Mead, 1965) [1]. Es un método basado en evaluaciones de la función objetivo. Por lo tanto no hace uso de información sobre el gradiente de la función (como en el caso de `fminunc`). En principio es adecuado para funciones que puedan presentar discontinuidades.

Sintaxis: `[x,fval,exitflag,output] = fminsearch(fun,x0,opciones)`
Argumentos y parámetros

`fun` = función objetivo a optimizar. Puede definirse en un M-archivo o mediante funciones anónimas.
`x0` = Semilla del algoritmo. Puede ser un escalar, un vector o una matriz.
`x` = mínimo local de `fun`.
`options` = opciones de optimización (ver `optimset`).
`fval` = devuelve el valor de la función objetivo `fun` en la solución x.
`exitflag` = describe la condición de salida de `fminsearch`.
`output` = Devuelve una estructura `output` que contiene optimización diversa sobre la optimización efectuada.

Ejercicio

Minimizar la función de Rosenbrock $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$, tomando como origen el punto $[-1.9, 2]$. Hacer una representación gráfica de la misma así como de los puntos generados.

---

fminunc

La función `fminunc` localiza el mínimo local de una función multivariante continua sin restricciones. Para ello, a diferencia de la anterior que se basa en la evaluación de la función objetivo, esta utiliza información del gradiente y el hessiano.

Sintaxis: `[x,fval,exitflag,grad,hessian] = fminunc (fun, x0, opciones)`
Argumentos y parámetros

- `x, fval:` similares al caso de `fminsearch`.
- `exitflag` (ojo, cambian los valores)
- `Grad`: Devuelve el gradiente de `fun` en `x`

- `Hessian`: Devuelve el hessiano de `fun` en `x`

Uso del gradiente y el hessiano

En la función `fminunc` Matlab diferencia dos tipos de algoritmos: de mediana escala y de gran escala. Entre los primeros se contemplan las versiones de Nelder y Mead (vista en `fminsearch`), BFGS (Broyden, Fletcher, Goldfard, Shanno) [2, 3, 4, 5] y DFP (Davidon, Fletcher y Powell) [6, 7, 8]. Para diferenciar ambos es necesario definir el parámetro `LargeScale`. Si se pone en `'off'` se usan los primeros. Si se pone en `'on'` se usan los segundos. En cualquiera de los dos casos la función `fminunc` admite que se defina el gradiente de la función objeto de estudio. En el caso de usar los métodos de gran escala es necesario definir dicho gradiente, siendo opcional en el caso de los de media escala.

La forma de definir el gradiente es mediante el parámetro `GradObj`. Por defecto está en `'off'` y no es necesario definirlo. Caso de que se desee hacerlo se hará como se muestra en el siguiente ejemplo.

Ejemplo:
Minimizar la función $f(x) = e^{x_1}\left(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1\right)$

Se crea el archivo objfungrad.m

```
function [f, G] = objfungrad(x)
% función objetivo
f=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);

% objfungrad tiene dos argumentos de salida
if nargout > 1  % calcula el gradiente
   t=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
   G=[ t + exp(x(1)) * (8*x(1) + 4*x(2)),
       exp(x(1))*(4*x(1)+4*x(2)+2)];
end
```

La forma de definir el hessiano es similar, con la salvedad de que hay que introducir un nuevo argumento de salida de la función y, además, en el código del M-archivo hay que definir el valor del hessiano haciendo una comprobación similar a la anterior sobre el número de argumentos de salida (`nargout` – ver documentación -).

Por defecto fminunc usa los métodos de gran escala si GradObj está en 'on'. Para la PARTE II de la práctica se debe dejar en 'off' con el fin de usar y comparar los métodos cuasi-Newton que proporciona Matlab.

En los métodos de media escala, en cada iteración es necesario realizar el cálculo de un hessiano para actualizar la dirección de búsqueda. Para ello es necesario invertir el hessiano del que se parte o, en su defecto, aproximarlo de algún modo. La función `fminunc` permite al usuario elegir la forma en que desea efectuar la actualización del hessiano a través del parámetro `HessUpdate`. Los valores que admite `HessUpdate` son:

- 'bfgs'

- 'dfp'
- 'steepdesc': escoge la dirección de descenso opuesta al gradiente. (No recomendada).

Como curiosidad, para ver un ejemplo de aplicación del método BFGS puede consultarse el siguiente enlace el cual muestra a través de un *applet* cómo evoluciona, gráfica y analíticamente, la búsqueda del mínimo de la función que se proporciona.
http://www.cse.uiuc.edu/eot/modules/optimization/BFGS/

Ejercicios: Minimizar las funciones siguientes haciendo uso de la función `fminunc` en sus diferentes variantes.

1. $f(x_1, x_2) = 2x_1^2 + 2x_1x_2 + x_2^2 + 2x_1 - 3$ (Solución: $(-1,1)$)
2. $f(x_1, x_2) = x_1x_2\left(x_1 - 1\right)$ (Solución: no tiene mínimos locales)
3. $f(x_1, x_2) = x_1x_2 \ln\left(x_1^2 + x_2^2\right)$ (Solución: $\left(\frac{1}{\sqrt{2e}}, \frac{1}{\sqrt{2e}}\right); \left(-\frac{1}{\sqrt{2e}}, -\frac{1}{\sqrt{2e}}\right)$)

Referencias

[1] Lagarias, J.C., J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal of Optimization*, Vol. 9, Number 1, pp.112-147, 1998.

[2] Broyden, C.G., "The Convergence of a Class of Double-Rank Minimization Algorithms," *Journal Inst. Math. Applic.*, Vol. 6, pp. 76-90, 1970.

[3] Fletcher, R.,"A New Approach to Variable Metric Algorithms," *Computer Journal*, Vol. 13, pp. 317-322, 1970.

[4] Goldfarb, D., "A Family of Variable Metric Updates Derived by Variational Means," *Mathematics of Computing*, Vol. 24, pp. 23-26, 1970.

[5] Shanno, D.F., "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computing*, Vol. 24, pp. 647-656, 1970.

[6] Davidon, W.C., "Variable Metric Method for Minimization," *A.E.C. Research and Development Report*, ANL-5990, 1959.

[7] Fletcher, R., "Practical Methods of Optimization," Vol. 1, *Unconstrained Optimization*, John Wiley and Sons, 1980.

[8] Fletcher, R. and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal*, Vol. 6, pp. 163-168, 1963.