Tema 4
Optimización no restringida en dimensión finita

Randall Romero Aguilar, PhD

Universidad de Costa Rica
SP6534 - Economía Computacional

I Semestre 2020

Última actualización: 10 de marzo de 2020

UCR
UNIVERSIDAD DE COSTA RICA

ESCUELA de ECONOMÍA
UNIVERSIDAD DE COSTA RICA

Tabla de contenidos

1. Teoría básica

Aplicaciones

Los problemas de optimización no restringida son ubicuos en economía:

- el gobierno maximiza el bienestar social
- el mercado competitivo maximiza el excedente total
- el estimador mínimos cuadrados ordinarios minimiza una suma de cuadrados
- el estimador de máxima verosimilitud maximiza la función de verosimilitud

Definiciones

- En el problema de optimización no restringida de dimensión finita buscamos, para una función $f : \Re^n \mapsto \Re$, un vector x^* tal que $f(x^*) \geq f(x)$ para todo x.
- A f la llamamos la función objetivo y a x^*, si existe, el máximo global de f.
- Nos enfocamos en maximización —para resolver problemas de minimización, simplemente maximizamos el negativo de la función objetivo.

Decimos que $x^* \in \Re^n$ es un...

- máximo global estricto de f si $f(x^*) > f(x)$ para todo $x \neq x^*$.
- máximo local de f si $f(x^*) \geq f(x)$ para todo x en algún vecindario de x^*.
- máximo local estricto de f si $f(x^*) > f(x)$ para todo $x \neq x^*$ en algún vecindario de x^*.

Condiciones de optimalidad

- Sea $f : \Re^n \mapsto \Re$ una función continuamente diferenciable dos veces.
- Condición necesaria de primer orden: Si x^* es un máximo local de f, entonces $f'(x^*) = 0$.
- Condición necesaria de segundo orden: Si x^* es un máximo local de f, entonces $f''(x^*)$ es semidefinida negativa.
- Decimos que x es un punto crítico de f si satisface la condición necesaria de primer orden.

- Condición suficiente: Si $f'(x^*) = 0$ y $f''(x^*)$ es definida negativa, entonces x^* es un máximo local estricto de f.
- Teorema local-global: Si f es cóncava, y x^* es un máximo local de f, entonces x^* es un máximo global de f.

Ejemplo 1:

Maximizando

$f(x) = x^3 - 12x^2 + 36x + 8$

- Consideremos la maximización de

$$f(x) = x^3 - 12x^2 + 36x + 8.$$

- La condición necesaria de primer orden

$$f'(x) = 3x^2 - 24x + 36 = 3(x-6)(x-2) = 0$$

- ...se satisface en los puntos críticos $x = 2$ y $x = 6$.

- Dado que

$$f''(x) = 6x - 24$$

se sigue que

$$f''(2) = -12 < 0 \quad \text{and} \quad f''(6) = 12 > 0$$

- Por lo tanto,
  - $x = 2$ satisface la condición suficiente para un máximo local estricto, pero
  - $x = 6$ no cumple la condición necesaria de segundo orden para un máximo local.

Ejemplo 2:

Maximizando

$$f(x) = 3 - x_1^2 - x_2^2 - x_1 x_2 + 2x_1 + x_2$$

- Consideremos la maximización de

$$f(x) = 3 - x_1^2 - x_2^2 - x_1 x_2 + 2x_1 + x_2.$$

- La condición necesaria de primer orden

$$f'(x) = \begin{bmatrix} -2x_1 - x_2 + 2 \\ -x_1 - 2x_2 + 1 \end{bmatrix} = 0$$

- ...se satisface en el punto crítico $x_1 = 1$ y $x_2 = 0$.

- El hessiano en el punto crítico

$$f''(x) = \begin{bmatrix} -2 & -1 \\ -1 & -2 \end{bmatrix}$$

tiene ecuación característica

$$\det \begin{bmatrix} -2-\lambda & -1 \\ -1 & -2-\lambda \end{bmatrix} = \lambda^2 + 4\lambda + 3 = (\lambda + 3)(\lambda + 1) = 0.$$

- El hessiano tiene eigenvalores negativos, -3 y -1, y por lo tanto es definida negativa.
- Entonces, $x = (1, 0)$ satisface la condición suficiente para un máximo local estricto.

Teorema de la envolvente

- El teorema de la envolvente nos dice cómo varía el máximo valor de una función con respecto a un parámetro.
- Sea $f : \Re^{n+1} \mapsto \Re$ una función real continuamente diferenciable. Si

$$V(\alpha) = \max_{x \in \Re^n} f(x, \alpha)$$

está bien definida y $x(\alpha)$ resuelve el problema de maximización, entonces

$$V'(\alpha) = \frac{\partial f(x(\alpha), \alpha)}{\partial \alpha}.$$

Ejemplo 3:
Teorema de la envolvente

- Si $f(x,\alpha) = \alpha x - 0.5x^2$, entonces

$$V(\alpha) \equiv \max_x f(x,\alpha) = 0.5\alpha^2$$

- Por lo tanto

$$V'(\alpha) = \alpha.$$

- Para cada α, el máximo es$x(\alpha) = \alpha$, así que, por el teorema de la envolvente,

$$V'(\alpha) = \frac{\partial f(x(\alpha),\alpha)}{\partial \alpha} = x(\alpha) = \alpha.$$

como se esperaba.

2. Algoritmos numéricos

Método de Newton-Raphson

- El método de Newton-Raphson maximiza un objetivo f usando sucesivas aproximaciones cuadráticas.
- Dada la k-ésima iteración x_k, la iteración subsiguiente x_{k+1} es calculada maximizando la aproximación cuadrática de f en x_k:

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}(x - x_k)' f''(x_k)(x - x_k).$$

- Resolviendo la condición de primer orden

$$f'(x_k) + f''(x_k)(x - x_k) = 0,$$

da por resultado la regla de iteración

$$x_{k+1} = x_k - [f''(x_k)]^{-1} f'(x_k).$$

- El método de Newton-Raphson es idéntico a usar el método de Newton para calcular la raíz del gradiente de la función objetivo.
- En teoría, convergirá si el valor inicial está cerca de un punto crítico de f en el cual el hessiano sea no-singular.
- En la práctica, divergirá si el valor inicial está lejos de un punto crítico o si el hessiano se vuelve mal condicionado.
- Además, puede converger a un punto crítico que no sea un máximo local, por lo que la condición necesaria de segundo orden siempre debe ser verificada.
- Newton-Raphson puede ser robusto al valor inicial si f es globalmente cóncava, pero sensible de lo contrario.

- Newton-Raphson tiene dos desventajas.
  1. requiere calcular tanto la primera como la segunda derivada.
  2. puede que no sea posible aumentar el objetivo en la dirección del paso de Newton...esto solo está garantizado si $f''(x_k)$ es definida negativa.
- Por esta razón, el método Newton-Raphson raramente es usado en la práctica, y solo si el objetivo es globalmente cóncavo.

Métodos cuasi-Newton

- En analogía con el método Newton-Raphson, los métodos cuasi-Newton actualizan las iteraciones en la dirección del vector

$$d_k = -A_k f'(x_k)$$

donde A_k es una aproximación de la hessiana inversa de f en la k-ésima iteración x_k.

- El vector d_k es conocido como el paso de Newton o cuasi-Newton.

- Similar a lo que sucede con problemas de búsqueda de raíces, no siempre es mejor dar un paso de Newton completo en cada iteración.
- Métodos cuasi-Newton eficiente acortan o alargan el paso de Newton para aumentar la ganancia en el objetivo.
- Esto se logra al ejecutar una búsqueda de línea en la cual el paso de Newton es re-escalado con un factor $s > 0$ que maximiza o casi maximiza $f\left(x_k + sd_k\right)$.
- Dado el factor de escala calculado s_k, actualizamos la iteración así:

$$x_{k+1} = x_k + s_k d_k.$$

- En la práctica, no es necesaria una búsqueda de línea exhaustiva.
- Por lo genaral, basta con aseguarse que la función objetivo aumente con cada iteración.
- Existen varios métodos de búsqueda de línea utilizados en la práctica.
- Tales métodos están más allá del alcance de este curso, pero son discutidos en la mayoría de textos de optimización aplicada.
- El paquete CompEcon ofrece cuatro métodos de búsqueda de línea.

- Los algoritmos cuasi-Newton difieren en cómo se construye y actualiza la aproximación de la hessiana inversaA_k.
- Algoritmos eficientes usan aproximaciones definidas negativas de la hessiana inversa, garantizando que el objetivo puede incrementarse en la dirección del paso de Newton.
- Algortimos cuasi-Newton eficientes además empleaan reglas de actualización que no requieren calcular segundas derivadas.
- El paquete CompEcon ofrece tres métodos de actualización.

- El método cuasi-Newton más simple fija $A_k = -I$, donde I es la matriz identidad, resultando en un paso de Newton idéntico al gradiente del objetivo:

$$d_k = f'(x_k).$$

- Este se llama el método de ascenso más empinado porque el gradiente, a un primer orden, promete el mayor incremento en f.
- El método de ascenso más empinado es simple, pero en la práctica menos eficiente numéricamente que métodos cuasi-Newton que utilizan información de la curvatura.

- Los métodos cuasi-Newton más ampliamente utilizados usan reglas de actualización del hessiano inverso que satisfacen dos condiciones.
- Primero, el hessiano inverso actualizado A_{k+1} debe satisfacer la condición cuasi-Newton:

$$x_{k+1} - x_k = A_{k+1}\left(f'(x_{k+1}) - f'(x_k)\right).$$

- Segundo, el hessiano inverso actualizado debe ser simétrico y definido negativo para asegurar que el objetivo puede incrementarse en la dirección del paso de Newton.
- Dos métodos de actualización que satisfacen las condiciones cuasi-Newton y de definida negativa son usadas en la práctica.

- El método Davidson-Fletcher-Powell (DFP) usa el esquema de actualización

$$A_{k+1} = A_k + \frac{v_k v_k'}{u_k' v_k} - \frac{A_k u_k u_k' A_k'}{u_k' A_k u_k},$$

donde

$$v_k = x_{k+1} - x_k$$

y

$$u_k = f'(x_{k+1}) - f'(x_k).$$

- El método Broyden-Fletcher-Goldfarb-Shano (BFGS) usa el esquema de actualización

$$A_{k+1} = A_k + \frac{1}{v_k' u_k}\left(w_k v_k' + v_k w_k' - \frac{u_k' w_k}{u_k' v_k} v_k v_k' \right),$$

donde

$$w_k = v_k - A_k u_k.$$

- BFGS en general supera a DFP, aunque hay problemas para los cuales DFP supera a BFGS.

- Los métodos cuasi-Newton son susceptibles a ciertos problemas.
- En ambas fórmulas de actualización hay una divisón por $v_k' u_k$.
- Si este valor llega a ser muy pequeño en valor absoluto, resultará en inestabilidad numérica.
- Por ello, es mejor omitir la actualización de A_k o reemplazarla con matriz identidad negativa reescalada si el valor se hace demasiado pequeño.

3. Ejemplos numéricos

La clase OP

- El paquete CompEcon tiene la clase `OP` (optimization problem) para calcular el máximo de una función $f : \Re^n \mapsto \Re$.
- Creamos un problema de optimización así:

```
from compecon import OP

def f(x):  #objective function
    return ... #function value

problem = OP(f)
x0 = ... #initial guess
x = problem.qnewton(x0) #local maximum of f
```

- Podemos escoger distintos métodos de actualización del hessiano inverso y de búsqueda de línea.

Ejemplo 4:

Máximo local de $x^3 - 12x^2 + 36x + 8$

- Para maximizar

$$f(x) = x^3 - 12x^2 + 36x + 8$$

empezando con $x = 4$, ejecutamos el código

```
F = OP(lambda x: x**3 - 12*x**2 + 36*x + 8)
x = F.qnewton(x0=4.0)
```

- Luego de 9 iteraciones obtenemos

```
x =  [2.]
```

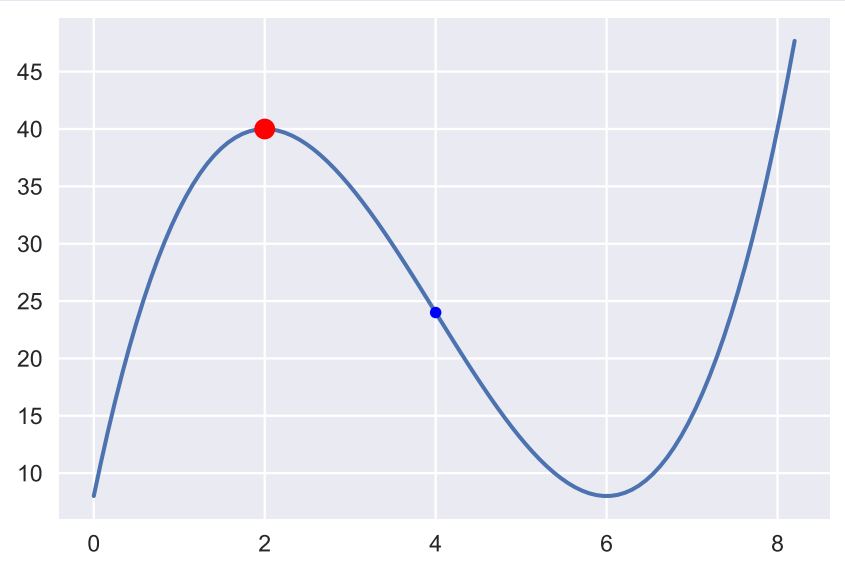

Figura 4.1: Función $f(x) = x^3 - 12x^2 + 36x + 8$

- Para ver la primera y segunda derivadas, ejecutamos

```
J = F.jacobian(x)
H = F.hessian(x)
E = np.linalg.eig(H)[0]
```

- lo que resulta en

```
J =  [-0.]
E =  [-12.]
```

- Por tanto, $x = 2$ es un máximo local estricto.

Ejemplo 5:

Máximo de

$$g(x,y) = 5 - 4x^2 - 2y^2 - 4xy - 2y$$

- Para maximizar

$$g(x,y) = 5 - 4x^2 - 2y^2 - 4xy - 2y$$

empezando con $x = (0,0)$, ejecutamos el código

```
def g(z):
    x, y = z
    return 5 - 4*x**2 - 2*y**2 - 4*x*y - 2*y

G = OP(g)
x = G.qnewton(x0=[-1, 1])
```

- Luego de 3 iteraciones obtenemos

```
x =  [ 0.5 -1. ]
```

- Para revisar el jacobiano y los eigenvalores del hessiano, ejecutamos

```
J = G.jacobian(x)
E = np.linalg.eig(G.hessian(x))[0]
```

- que da por resultado

```
J =  [0. 0.]
E =  [-10.4721  -1.5279]
```

- Por lo tanto, $x = (0.5, -1.0)$ es un máximo local estricto.

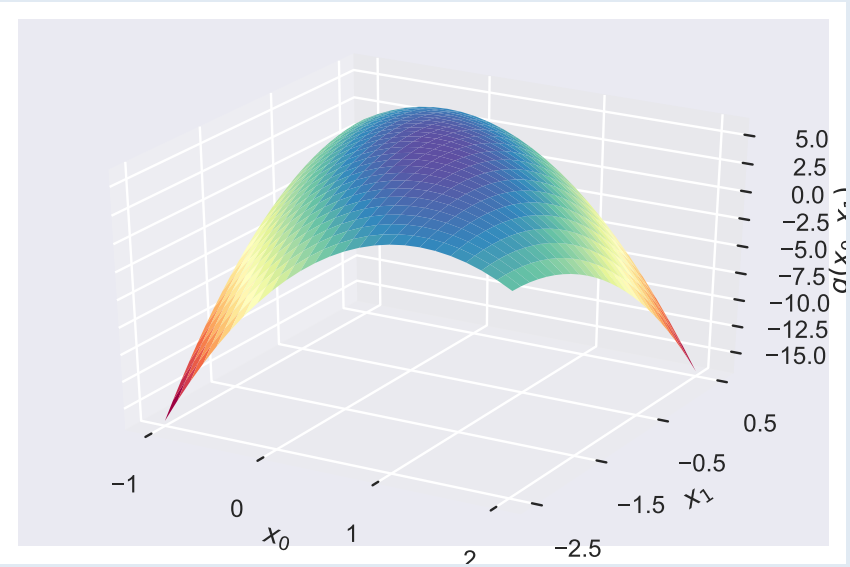

Figura 4.2: Función $g(x, y) = 5 - 4x^2 - 2y^2 - 4xy - 2y$

Ejemplo 6:
Maximizando la función de Rosencrantz

- Para maximizar la función de Rosencrantz o función banano

$$f(x,y) = -100(y-x^2)^2 - (1-x)^2$$

empezando con $x_0 = (1,0)$, ejecutamos el código

```
def f(z):
    x, y = z
    return -100 * (y - x**2)**2 - (1 - x)**2

x0 = [1, 0]
banana = OP(f)
x = banana.qnewton(x0)
```

- Luego de 27 iteraciones obtenemos

```
x = [1. 1.]
```

- Para revisar el jacobiano y los eigenvalores del hessiano, ejecutamos

```
J = banana.jacobian(x)
E = np.linalg.eig(banana.hessian(x))[0]
```

- para obtener

```
J =  [-0.  0.]
E =  [-1001.6006     -0.3994]
```

- Por lo tanto, $x = (1, 1)$ es un máximo local estricto.

- Para maximizar la función con otros métodos, podemos sobrescribir el método predeterminado así:

```
banana.qnewton(x0, SearchMeth='steepest')
banana.qnewton(x0, SearchMeth='bfgs')
banana.qnewton(x0, SearchMeth='dfp')
```

- 'steepest' no logra encontrar el óptimo en 250 iteraciones, el máximo predeterminado. Los senderos de búsqueda son:

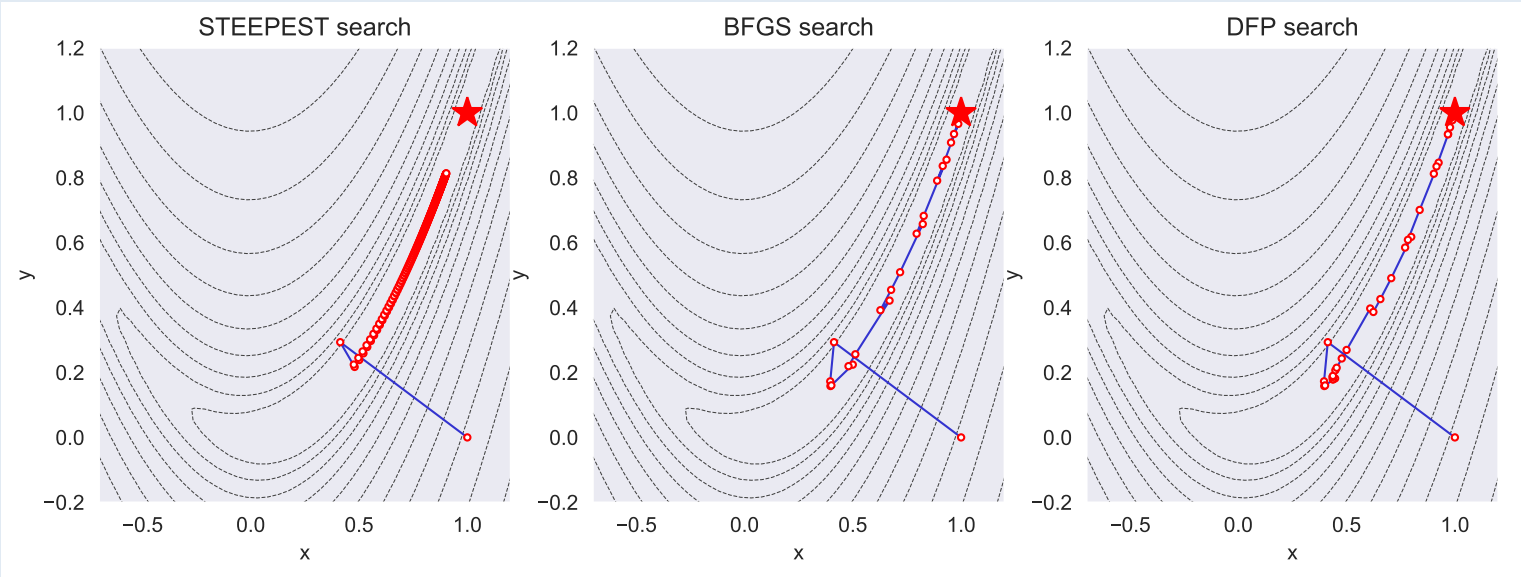

Figura 4.3: Maximización de la función de Rosencrantz

4. Casos especiales

- Dos clases especiales de problemas de optimización son frecuentes en econometría y merecen una discusión adicional
- Mínimos cuadrados no lineales y máxima verosimilitud tienen estructuras especiales que dan pie a métodos cuasi-Newton eficientes que usan aproximaciones distintas a la hessiana inversa.
- Como estos problemas usualmente aparecen en aplicaciones estadísticas, cambiamos nuestra notación para seguir las convenciones en esas aplicaciones.
- La optimización se realiza con respecto al vector de parámetros θ de dimensión k, y n se refiere al número de observaciones.

Mínimos cuadrados no lineales

- El problema de mínimos cuadrados no lineales tiene la forma

$$\min_{\theta} \tfrac{1}{2} f(\theta)^\top f(\theta) \quad = \quad \min_{\theta} \sum_{i=i}^{n} \tfrac{1}{2} f_i^2(\theta)$$

  donde $f : \Re^k \to \Re^n$.
- La función objetivo tiene gradiente

$$\sum_{i=1}^{n} f_i'(\theta) f_i(\theta) = f'(\theta)^\top f(\theta)$$

  y hessiano

$$f'(\theta)^\top f'(\theta) + \sum_{i=1}^{n} f_i(\theta) \frac{\partial^2 f(\theta)}{\partial \theta \partial \theta^\top}.$$

- Ignorando el segundo término del hessiano da por resultado una matriz definida positiva con la cual determinar la dirección de búsqueda:

$$d = -\left[f'(\theta)^\top f'(\theta) \right]^{-1} f'(\theta)^\top f(\theta).$$

Ejemplo 7:
Estimación mínimos cuadrados no lineales

Greene (2012, p.191) considera la siguiente función de consumo no lineal

$$C = \alpha + \beta * Y^{\gamma} + \epsilon$$

la cual es estimada con datos trimestrales de consumo e ingreso disponible real de la economía de EE.UU. de 1950 a 2000.

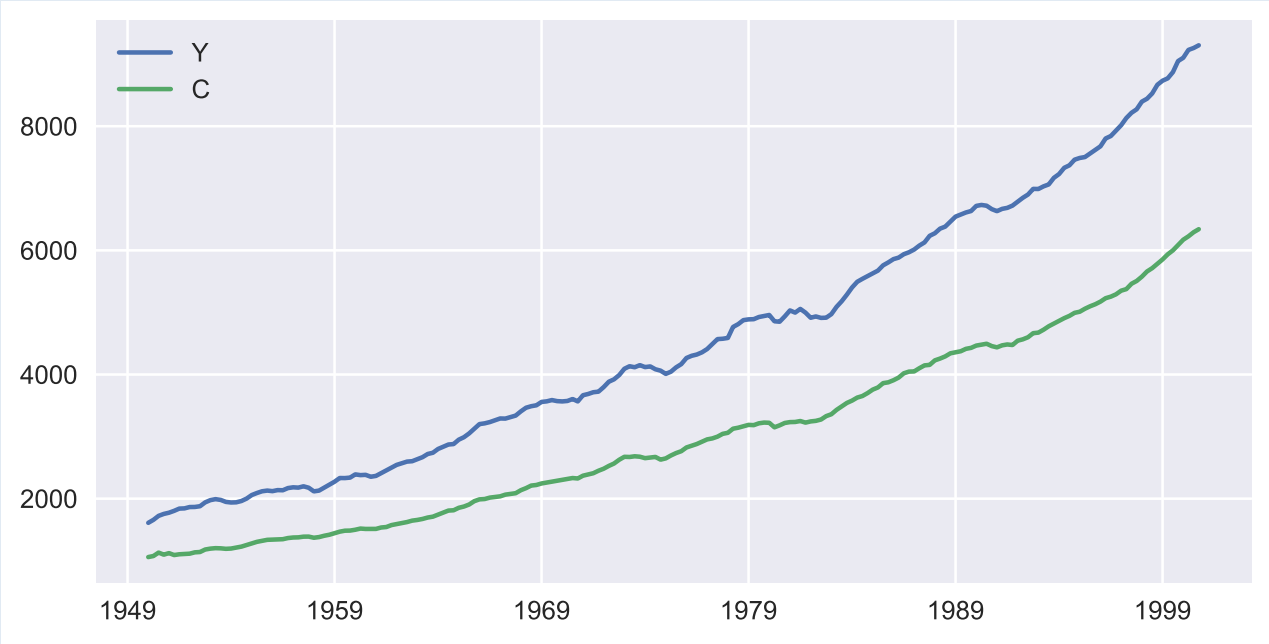

Figura 4.4: Ingreso y consumo en EE.UU.

Para obtener los datos

```
import pandas as pd
data = pd.read_table('TableF5-2.txt',sep='\s+')
Y, C = data[['realgdp','realcons']].values.T
```

La función objetivo es el negativo de la suma de los residuos al cuadrado

```
def ssr():
    , ,  =
    residuals = C -  - *Y**
    return -(residuals**2).sum()
```

Encontramos el estimador mínimos cuadrados no lineales, partiendo del valor $(\alpha, \beta, \gamma) = (0, 0, 1)$

```
from compecon import OP
nlls = OP(ssr).qnewton([0.0, 0.0, 1.0])
```

Esto da por resultado $(\alpha, \beta, \gamma) =$

```
[-91.1965    0.5691    1.0204]
```

Este resultado no es igual al reportado en el libro de Greene, pero puede ser reproducido con Stata:

```
import delimited TableF5-2.txt, delimiter(space, collapse)
nl (realcons = {alpha=0.0} + {beta=0.0}*realgdp ^{gamma=1.0})
```

Estimación de máxima verosimilitud

- Uno observa una muestra aleatoria y_i tomada de una distribución con f.d.p. $f(y;\theta)$ donde θ es un parámetro no observado.
- El estimador de máxima verosimilitud de θ maximiza la verosimilitud muestral

$$L(\theta;y) = \prod_{i=1}^{n} f(y_i;\theta).$$

o, equivalentemente, maximiza la log-verosimilitud muestral

$$LL(\theta;y) = \sum_{i=1}^{n} \ln f(y_i;\theta).$$

- Los problemas de máxima verosimilitud se especifican escogiendo una función de distribución f para los datos y que depende de un vector de parámetros θ.
- La función log-verosimilitud es la suma de los logaritmos de las verosimilitudes de cada observación puntual:

- Un resultado muy conocido en teoría estadística dice que la esperanza del producto interior de la función *score* es igual al negativo de la esperanza de la hessiana de la función de verosimilitud.
- Por ello, el promedio muestral del producto interior de la función *score* es una aproximación razonable de un hessiano definido positivo que puede usarse para determinar la dirección de búsqueda:

$$d = -\left[s(\theta; y)^\top s(\theta, y)\right]^{-1} s(\theta, y)' 1_n,$$

  donde 1_n es un n-vector de unos.
- Este enfoque se conoce como el método modificado de *scoring*.

Ejemplo 8:
Estimación de máxima verosimilitud

Greene (2012, p.590) considera el siguiente modelo de escogencia binaria

$$\mathbb{P}[\text{GRADE} = 1] = F(\beta_0 + \beta_1 \text{GPA} + \beta_2 \text{TUCE} + \beta_3 \text{PSI})$$

donde F es la función de distribución acumulada de la distribución normal (probit) o logística (logit).

Para obtener los datos, así como la f.d.a. para las distribuciones normal y logística:

```
from scipy.stats import norm, logistic

data = pd.read_table('TableF14-1.txt',sep='\s+')
data['intercept'] = 1
regressors = ['intercept', 'GPA','TUCE','PSI']

X = data[regressors]
y = data['GRADE']
```

La función log-verosimilitud de un modelo de escogencia binaria viene dadad por

$$\ln L = \sum_{i=1}^{n} \left\{ y_i \ln F(x_i'\beta) + (1 - y_i) \ln[1 - F(x_i'\beta)] \right\}$$

la cual programamos así

```
def binary_model(,distribution):
    F = distribution.cdf(X @ )
    return (y*np.log(F) + (1-y)*np.log(1-F)).sum()

def logL_logit():
    return binary_model(,logistic)

def logL_probit():
    return binary_model(,norm)
```

Entonces estimamos el modelo

```
0 = np.zeros(4) # initial guess

_logit = OP(logL_logit).qnewton(0, SearchMeth='bfgs')
_probit = OP(logL_probit).qnewton(_logit/2, SearchMeth='bfgs')

pd.DataFrame({'logit':_logit,'probit':_probit},
             index=regressors)
```

lo que da por resultado

```
             logit  probit
intercept -13.021  -7.452
GPA         2.826   1.626
TUCE        0.095   0.052
PSI         2.379   1.426
```

Estos resultados pueden replicarse con Stata:

```
infix obs 1-3 gpa 10-14 tuce 19-23 psi 28 grade 37...
            using TableF14-1.txt in 2/33

logit grade gpa tuce psi
probit grade gpa tuce psi
```

References I

- Greene, William H. (2012). *Econometric Analysis*. 7ª ed. Prentice Hall. isbn: 978-0-13-139538-1.
- Miranda, Mario J. y Paul L. Fackler (2002). *Applied Computational Economics and Finance*. MIT Press. isbn: 0-262-13420-9.
- Romero-Aguilar, Randall (2016). *CompEcon-Python*. url: http://randall-romero.com/code/compecon/.