1 / 1 punto

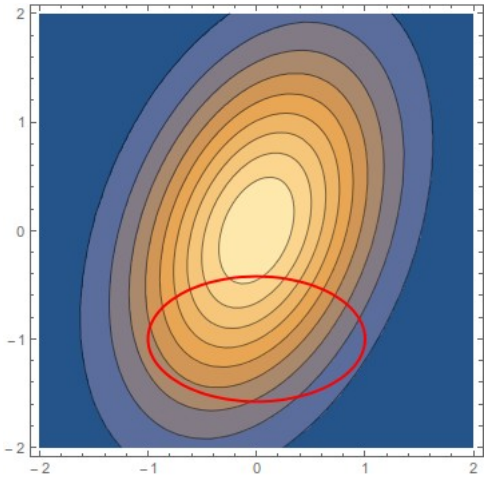
Consideremos el ejemplo de encontrar el mínimo de la función,

$$f(\mathbf{x}) = \text{Exp}\left(-\frac{2x\_^2 + y^2 - xy}{2}\right)$$

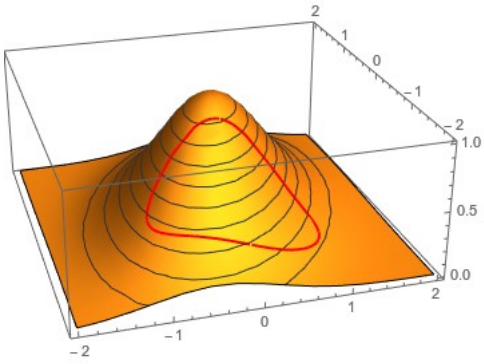
a lo largo de la curva (o, sujeto a la restricción),

$$gramo(\mathbf{x}) = X^2 + 3(y+1)^2 - 1 = 0\,.$$

Las funciones en sí son bastante simples, en un mapa de contorno se ven de la siguiente manera,



Sin embargo, su combinación puede volverse bastante complicada si se calcularan directamente, como se puede inferir de la forma de la restricción en el gráfico de superficie,



Tenga en cuenta, en este caso, la función $f(\mathbf{x})$ no tiene mínimos en sí, pero a lo largo de la curva hay dos mínimos (y dos máximos).

Una situación como esta es donde entran en juego los multiplicadores de Lagrange. La observación es que los máximos y mínimos de la curva se encontrarán donde la restricción es paralela a los contornos de la función.

Como el gradiente es perpendicular a los contornos, el gradiente de la función y el gradiente de la restricción también serán paralelos, es decir,

$$\nabla f(\mathbf{x}) = \lambda\ \nabla gramo(\mathbf{x})$$

Si escribimos esto en forma de componentes, esto se convierte en,

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = yo \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix}$$

Esta ecuación, junto con $gramo(\mathbf{x}) = 0$ es suficiente para especificar el sistema completamente. Podemos poner toda esta información en una sola ecuación vectorial,

$$\nabla \mathrm{L}\left(x, y, yo\right) = \begin{bmatrix} \frac{\partial f}{\partial x} - yo\frac{\partial g}{\partial x} \\ \frac{\partial f}{\partial y} - yo\frac{\partial g}{\partial y} \\ -\,gramo\left(\mathbf{x}\right) \end{bmatrix} = 0$$

y luego resuelve esta ecuación para resolver el sistema.

Reflexionemos sobre lo que hemos hecho aquí, hemos pasado de encontrar un mínimo de una función 2D restringida a una curva 1D, a encontrar los ceros de una ecuación vectorial 3D.

Si bien esto puede sonar como si hubiéramos hecho el problema más complicado, estamos exactamente equipados para enfrentar este tipo de problema; podemos usar los métodos de búsqueda de raíces, como el método de Newton-Raphson que hemos discutido anteriormente.

Vamos a configurar el sistema,

La función y dos de las derivadas están definidas para ti. Configure los otros dos reemplazando los signos de interrogación en el siguiente código.

```
# Primero definimos las funciones,
definición  f (x, y) :
    return  np.exp(-( 2 *x*x + y*y - x*y) /  2 )

definición  g (x, y) :
    devuelve  x*x +  3 *(y+ 1 )** 2  -  1

# Luego sus derivadas,
def  dfdx (x, y) :
    devuelve 1/2 *  ( -4 *x + y) * f(x, y )

def  dfdy (x, y) :
    devuelve 1 / 2  * (x -  2 *y) * f(x, y)

definición  dgdx (x, y) :
    volver 2 *x

definición  dgdy (x, y) :
    return 6*(y+1)
```

Ejecutar

Restablecer

Correcto

Bien hecho.

2. A continuación, definamos el vector $\nabla \mathrm{L}$, que vamos a encontrar los ceros de; llamaremos a esto "DL" en el código. Entonces podemos usar un método de búsqueda de raíces preescrito en scipy para resolver. **1 / 1 punto**

```
de  scipy  importar  optimizar

definición  DL (xyλ) :
    [x, y, λ] = xyλ
    volver  np.matriz([
            dfdx(x, y) - λ * dgdx(x, y),
            dfdy(x, y) - λ * dgdy(x, y),
            - g(x, y)
        ])

(x0, y0, λ 0 ) = ( -1 ,  -1 ,  0 )
x, y, λ = optimizar.root(DL, [x0, y0, λ 0 ]).x
imprimir ( "x = %g"  % x)
imprimir ( "y = %g"  % y)
imprimir ( "λ = %g"  % λ)
imprimir ( "f(x, y) = %g"  % f(x, y))
```

Ejecutar

Restablecer

Aquí, los dos primeros elementos de la matriz son los X y coordenadas que queríamos encontrar, y el último elemento es el multiplicador de Lagrange, que podemos tirar ahora que se ha utilizado.

Mira esto (X , y) efectivamente resuelve la ecuación $g\left(x, y\right) = 0$.

Debería poder usar el código para encontrar las otras raíces del sistema.

Reutilice el código anterior con diferentes valores iniciales para encontrar los otros puntos estacionarios en la restricción.

Hay cuatro en total. Dar el $y$coordenada de cualquiera de las otras soluciones con dos decimales.

-0.43

Correcto
Este es uno de los máximos o mínimos.

3. En la pregunta anterior, usted dio la $y$coordenada de *cualquiera* de los puntos estacionarios. En esta parte, da la Xcoordenada del *mínimo global* de $f(x)$ en $gramo(x) = 0$. **1 / 1 punto**

Da tu respuesta con 2 decimales.

0.93

Correcto
Efectivamente, este es el mínimo global, sujeto a la restricción.

4. Tal vez se pregunte por qué el vector ∇L obtiene un símbolo divertido. Esto se debe a que se puede escribir como el gradiente (sobre X, y , yyo) de una función escalar $L(x, y, yo)$. **1 / 1 punto**

Con base en su conocimiento de las derivadas, ¿qué función $L(x, y, yo)$ daría la forma esperada de ∇L?

- $L(x, y, yo) = f(\mathbf{x}) - \lambda\, gramo(\mathbf{x})$
- $L(x, y, yo) = f(\mathbf{x}) + gramo(\mathbf{x}) + yo$
- $L(x, y, yo) = f(\mathbf{x}) + \lambda\, gramo(\mathbf{x}) + yo$
- $L(x, y, yo) = f(x, y) + g(x, y, yo)$

Correcto
Exactamente, esta es también la razón por la cual el último componente de ∇L tiene un signo menos.

5. Esperemos que ahora haya desarrollado una idea de cómo funcionan los multiplicadores de Lagrange. Probemos esto en una nueva función y restricción. **1 / 1 punto**

Calcular el mínimo de

$$f(x, y) = -\exp(x - y^2 + x\,y)$$

en la restricción,

$$g(x, y) = \text{Cosh}(y) + X - 2 = 0$$

Usa el código que has escrito en las preguntas anteriores para ayudarte.

```
7       volver  -np.exp(x - y*y + x*y)
8
9   definición  g (x, y) :
10      devuelve  np.cosh(y) + x -  2
11
12  # A continuación de sus derivados, DEBE IMPLEMENTAR EL SE
13  def  dfdx (x, y) :
14      retorno  -( 1 +y)*np exp(x - y*y + x*y)
```

```
    retorno  -( 1 +y) np.exp(x - y y + x y)

def  dfdy (x, y) :
    retorno  ( 2 *yx)*np.exp(x - y*y + x*y)

definición  dgdx (x, y) :
    volver 1

definición  dgdy (x, y) :
    return  np.sinh(y)

# Use la definición de DL de anteriormente.
definición  DL (xyλ) :
    [x, y, λ] = xyλ
    volver  np.matriz([
            dfdx(x, y) - λ * dgdx(x, y),
            dfdy(x, y) - λ * dgdy(x, y),
            - g(x, y)
        ])

# Para puntuar en esta pregunta, el código anterior debe  establecer
# las variables x, y, λ, a los valores que resuelven  el
# Problema del multiplicador de largo alcance.

# Es decir, use el método de optimización.root, como lo hizo anteriormente.

x, y, λ = optimizar.root(DL, [ 0 ,  0 ,  0 ]).x

imprimir ( "x = %g"  % x)
imprimir ( "y = %g"  % y)
imprimir ( "λ = %g"  % λ)
imprimir ( "f(x, y) = %g"  % f(x, y))
```

Ejecutar

Restablecer

```
x = 0,957782
y = 0.289565
λ = -4.07789
f(x, y) = -3.16222
```

Correcto

Bien hecho. Ha construido y ejecutado su propio solucionador de multiplicadores de Lagrange para una nueva función.