Proyecto Abril: "Optimización"
MAT1189

Yanina Baeza
Fernando Ferrari
Sofia Herrera
Magdiel Soto

Abril 2021

1. Introducción

Optimización hace referencia a la acción y efecto de optimizar. En términos generales, se refiere a la capacidad de hacer o resolver alguna cosa de la manera más eficiente posible y, en el mejor de los casos, utilizando la menor cantidad de recursos. En el caso más simple, un problema de optimización consiste en maximizar o minimizar una función real eligiendo sistemáticamente valores de entrada y computando el valor de la función.

La optimización en informática es el proceso que busca mejorar el rendimiento del software, hardware o redes de un sistema para que funcione de manera eficiente. En la optimización del hardware, entran todos los elementos externos de una terminal, incluyendo los periféricos. Estos pueden ser modificados o cambiados para mejorar el rendimiento del equipo, pero también pueden tener un objetivo estético. Por su parte, la optimización de software busca adaptar un programa para que funcione mejor. En este sentido, la eficiencia de la optimización tiene que ver con mejoras en la velocidad, cantidad de memoria utilizada, tiempo de ejecución, uso de ancho de banda y consumo de energía.

2. Modelo

Modelo 1 de Weibull (simplificado):

$$u(t) = 1 - e^{-at}$$

Modelo 2 de Korsmeyer-Peppas:

$$u(t) = at^b$$

Función de Costo:

$$f(p) = \sum_{i=1}^{n} (y(p; x_i) - \hat{y}_i)^2$$

2.1. Cálculo de función con una variable:

Condición necesaria:

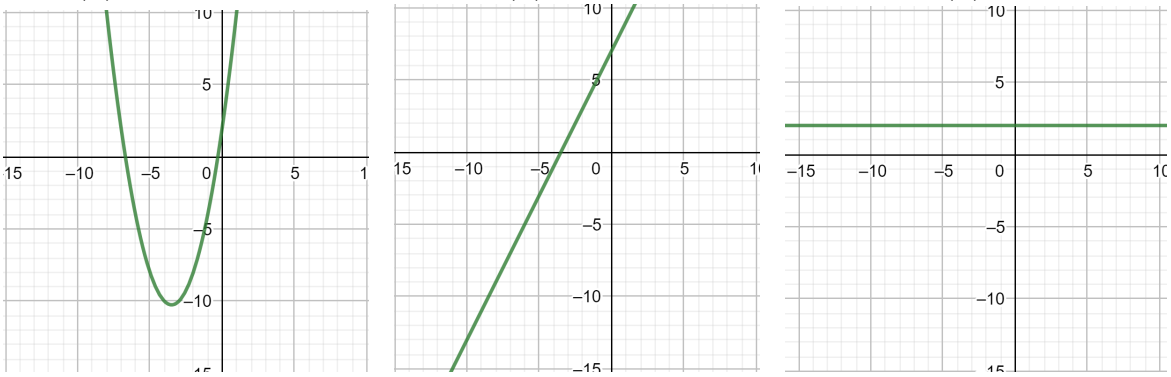
$$f'(x) = \frac{\partial}{\partial x} f(x) = 0$$

Condición suficiente:

$$f''(x) = \frac{\partial^2}{\partial x^2} f(x) \neq 0$$

Ejemplo:

$f(x) = x^2 + 7x + 2$ $\quad$ $f'(x) = 2x + 7$ $\quad$ $f''(x) = 2$



2.2. Cálculo función de costo con dos variables:

La condición necesaria para un optimo es que el gradiente sea igual a cero en cada uno de sus componentes. Y la condición suficiente es que la matriz Hessiana está definida, positiva definida para un mínimo.

Condición necesaria:

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial}{\partial x} f(x, y) \\ \frac{\partial}{\partial y} f(x, y) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Condición suficiente:

$$\begin{pmatrix} q_1 & q_2 \end{pmatrix} H(x, y) \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} > 0 \qquad \forall q_1, q_2$$

$$q_1^2 f_{xx} + 2 q_1 q_2 f_{xy} + q_2^2 f_{yy} > 0 \qquad \forall q_1, q_2$$

2.3. Matriz Hessiana

La matriz Hessiana es la generalización de la segunda derivada. La matriz Hessiana es simétrica.

$$H(x,y) = \begin{pmatrix} \partial_{xx} f(x,y) & \partial_{xy} f(x,y) \\ \\ \partial_{xy} f(x,y) & \partial_{xy} f(x,y) \end{pmatrix}$$

$$H(x,y) = \begin{pmatrix} \dfrac{\partial^2}{\partial x^2} f(x,y) & \dfrac{\partial^2}{\partial xy} f(x,y) \\ \dfrac{\partial^2}{\partial xy} f(x,y) & \dfrac{\partial^2}{\partial y^2} f(x,y) \\ \end{pmatrix}$$

Ejemplo:

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

- Determinar el gradiente de la función:

$$\nabla f(x,y) = (2x + y - 5) \cdot i + (2y + x - 4) \cdot j$$

- Matriz Hessiana:

$$\begin{array}{ll} f_x = 2x + y - 5 & f_{xx} = 2 \\ & f_{xy} = 1 \\ f_y = 2y + x - 4 & f_{yx} = 1 \\ & f_{yy} = 2 \end{array}$$

$$\mathrm{H(x,y)} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

- Autovalores de la matriz Hessiana:

$$H - \lambda \cdot I$$

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}$$

$$\begin{pmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{pmatrix}$$

$$(2 - \lambda)^2 - 1 = 0$$

$$\lambda^2 - 4\lambda + 3 = 0$$

$$\lambda_1 = 1 \quad ; \quad \lambda_2 = 3$$

2.4. Calcular el gradiente de la función de costo del modelo 2

$$\sum_{i=1}^{n}(at^b - \hat{y}_i)^2$$

$$\nabla f(a,b) = [2(at^b - \hat{y}_i)t^b] + [2(at^b - \hat{y}_i])at]$$
$$\nabla f(a,b) = [(2at^b - 2\hat{y}_i)t^b] + [(2at^b - 2\hat{y}_i)at]$$
$$\nabla f(a,b) = (2at^{2b} - 2\hat{y}_i t^b)\cdot i + (2a^2t^{b+1} - 2\hat{y}_i at)\cdot j$$

2.5. Calcular la matriz Hessiana de la función de costo de modelo 2

$$f_a = 2at^{2b} - 2\hat{y}_i t^b \qquad f_{aa} = 2t^{2b}$$
$$f_{ab} = 2at^2 - 2\hat{y}_i t$$
$$f_b = 2a^2t^{b+1} - 2\hat{y}_i at \qquad f_{ba} = 4at^{b+1} - 2\hat{y}_i t$$
$$f_{bb} = 2a^2t^2$$

$$H(a,b) = \begin{pmatrix} 2t^{2b} & 2at^2 - 2\hat{y}_i t \\ 4at^{b+1} - 2\hat{y}_i t & 2a^2t^2 \end{pmatrix}$$

2.6. Aproximación por función cuadrática

Una función vectorial no-lineal puede ser aproximada por una función cuadrática. La matriz A de esta aproximación coincide con la matriz Hessiana, el vector b con el gradiente y el escalar c con la evaluación de la función, en el punto en el cual se realiza la aproximación.

$$A = H(x_0, y_0),\ b = \nabla f(x_0, y_0),\ c = f(x_0, y_0)$$

La función general de aproximación cuadrática es la siguiente:

$$Q(x,y) = f(x_0,y_0) + f_x(x_0,y_o)(x - x_0) + f_y(x_0,y_0)(y - y_0) + \frac{1}{2}f_{xx}(x_0,y_0)(x - x_0)^2 +$$

$$f_{xy}(x_0,y_0)(x - x_0)(y - y_0) + \frac{1}{2}f_{yy}(x_0,y_0)(y - y_0)^2$$

Ejemplo:

$$f(x,y) = e^{\frac{x}{2}}sen(y)$$

Buscamos las derivadas parciales que nos pide la fórmula de la función $f(x,y)$

$$f_x(x,y) = \frac{1}{2}e^{\frac{x}{2}}sen(y)$$

$$f_y(x,y) = e^{\frac{x}{2}}cos(y)$$
$$f_{xx}(x,y) = \frac{1}{4}e^{\frac{x}{2}}sen(y)$$
$$f_{xy}(x,y) = \frac{1}{2}e^{\frac{x}{2}}cos(y)$$
$$f_{yy}(x,y) = -e^{\frac{x}{2}}sen(y)$$

Ahora evaluaremos la función en un punto, para el ejemplo elegimos el punto $P(0, \frac{\pi}{2})$

$$\begin{array}{ll} f(x,y)=1 & f_x(x,y)=\frac{1}{2} \\ f_y(x,y)=0 & f_{xx}(x,y)=\frac{1}{4} \\ f_{xy}(x,y)=0 & f_{yy}(x,y)=-1 \end{array}$$

Lo siguiente es reemplazar los valores obtenidos en la fórmula.

$$Q(x,y)=1+\frac{1}{2}(x-0)+0\cdot(y-\frac{\pi}{2})+\frac{1}{2}\cdot\frac{1}{4}(x-0)^2+0\cdot(x-0)(y-\frac{\pi}{2})+\frac{1}{2}(-1)(y-\frac{\pi}{2})^2$$

$$Q(x,y)=1+\frac{x}{2}+\frac{x^2}{8}-\frac{1}{2}(y-\frac{\pi}{2})^2$$

2.7. Función Convexa

Un concepto básico para caracterizar una función en cuanto a sus óptimos es la convexidad. Veamos la definición de la convexidad.

Una función $f: I \to \mathbb{R}$ se llama convexa si para cualquier $x, y \in I$ $s \in [0,1]$ tenemos

$$f(sx+(1-s)y) \leq sf(x)+(1-s)f(y).$$

Ejemplo:

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

$$\nabla f=[2x+y^2-1]+[2xy]$$

$$f_x=2x+y^{-}1 \quad \begin{array}{c} f_{xx}=2 \\ f_{xy}=2y \\ f_{yx}=2y \\ f_{yy_{=}2x} \end{array} \quad \text{H(x,y)=} \begin{pmatrix} 2 & 2y \\ 2y & 2x \end{pmatrix}$$

$$f_y=2xy$$

$$D_1=2<0$$

$$D_2=\begin{vmatrix} 2 & 2y \\ 2y & 2x \end{vmatrix}$$

$$D_2=4x-4y^2$$

$$\begin{Bmatrix} D_2>0 => & EstrictamenteConvexa \\ D_2=0 => & Convexa \\ D_2<0 => & Indefinida \end{Bmatrix}$$

3. Algoritmos

3.1. Algoritmo de Nelder-Mead(fminsearch)

El método de Nelder-Mead es ampliamente utilizado en química, ingeniería química y medicina. Además, desde su publicación se ha convertido en uno de los métodos más usados en programación no lineal sin restricciones.

En Octave, fminsearch es un método que se usa para encontrar el valor de x que minimice la función. Se recomienda usar fminsearch para funciones con discontinuidades , o cuando la búsqueda de la gradiente falla.

```
x = fminsearch (fun, x0, options)
```

fminsearch minimiza la funcion fun con los parámetros de optimización especificados en options. La busqueda comienza en x0 e itera usando el algoritmo Nelder-Mead Simplex.

3.2. Rutina fminunc(Octave)

Muchas veces es útil encontrar el valor minimo de una función en vez de donde se cruzan los ceros con el eje x. Se recomienda usar fminunc para funciones diferenciales y problemas de optimización sin restricciones.

```
fminunc (fcn, x0, options)
```

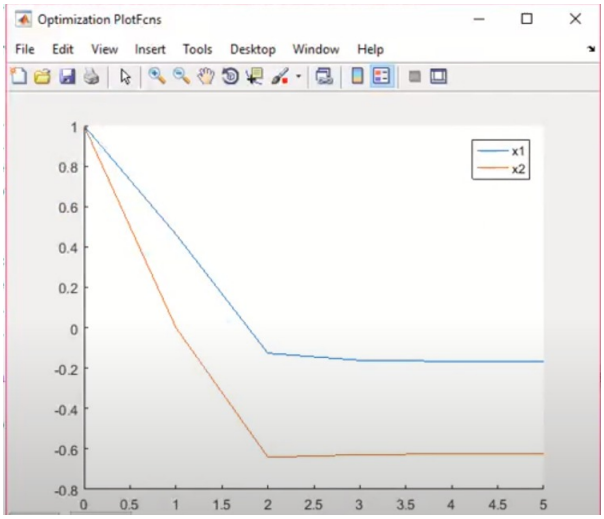
El parámetro fcn acepta un arreglo de variables y devuelve el valor de la función objetivo, opcionalmente con gradiente.

```
##
function j = funFminunc(x)
n=2; % variables del problema
x1=x(1);
x2=x(2);

%condiciones
PM=0;
% calculos previos
Valor=0;

% funcion objetivo
j =  3*x1^2+4*x2^2+x1+5*x2+7
```

Se describe los parámetros de la función junto con la función objetivo

```
x0=[1 1]';

opciones=optimoptions('fminunc','Algorithm','quasi-newton','Display','iter','Plotfcn',{@optimplotxval});

% intrucciones de resolucion

x = fminunc(@funFminunc,x0,opciones)
```

Script en el cual se ejecuta la Fminunc



Vista gráfica en tiempo real de la ejecución del Script

Iteration	Func-count	f(x)	Step-size	First-order optimality
0	3	20		13
1	6	8.10059	0.0769231	5
2	9	5.3601	1	0.242
3	12	5.35431	1	0.0323
4	15	5.35417	1	0.000143
5	18	5.35417	1	1.31e-06

Local minimum found.

Optimization completed because the size of the gradient is less than the default value of the optimality tolerance.

<stopping criteria details>

```
x =

   -0.1667
   -0.6250


y =

    5.3542
```

Resultado de la ejecución en la consola con y como el valor mínimo

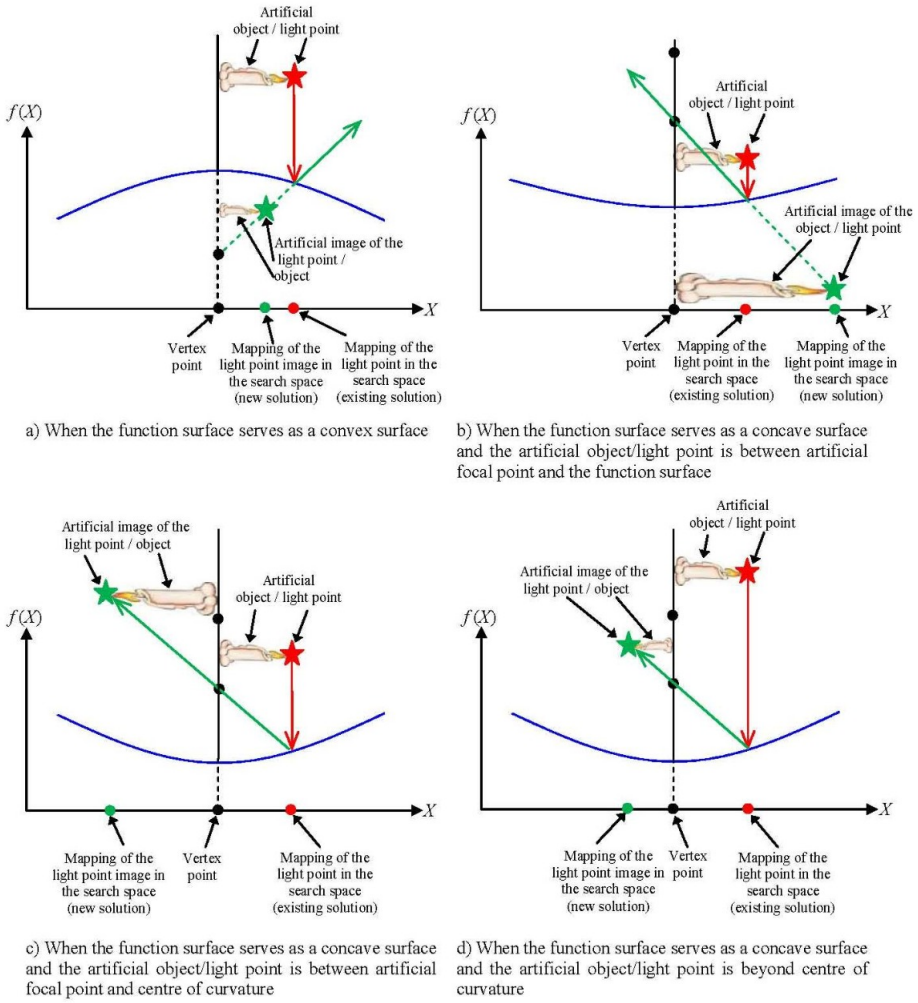
3.3. Método de Gauss-Newton

El algoritmo de Gauss-Newton se utiliza para resolver problemas no lineales de mínimos cuadrados. Es un procedimiento iterativo. Esto significa que debemos proporcionar una estimación inicial del parámetro vector

3.4. O.I.O (Optics Inspired Optimization)

Optics Inspired Optimization (OIO) es un algoritmo evolutivo inspirado en la óptica en el que se supone que una serie de puntos de luz artificial (puntos en $Rn + 1$ cuyo mapeo en Rn son posibles soluciones al problema) se encuentran frente a un dispositivo artificial. espejo ondulado que refleja sus imágenes. OIO trata la superficie de la función a optimizar como el espejo reflectante compuesto por picos y valles. Cada pico se trata como una superficie reflectante convexa y cada valle se trata como una superficie reflectante cóncava. De esta manera, el rayo artificial que brilla desde el punto de luz artificial es reflejado artificialmente por la superficie funcional, dado que la superficie reflectante es parte de un pico o parte de un valle, y el punto de imagen artificial (un nuevo punto en $Rn+1$ que está mapeado en

Rn como una nueva solución en el dominio de búsqueda) se forma en posición vertical (hacia la posición del punto de luz en el espacio de búsqueda) o invertido (hacia afuera de la posición del punto de luz en el espacio de búsqueda).



a) When the function surface serves as a convex surface

b) When the function surface serves as a concave surface and the artificial object/light point is between artificial focal point and the function surface

c) When the function surface serves as a concave surface and the artificial object/light point is between artificial focal point and centre of curvature

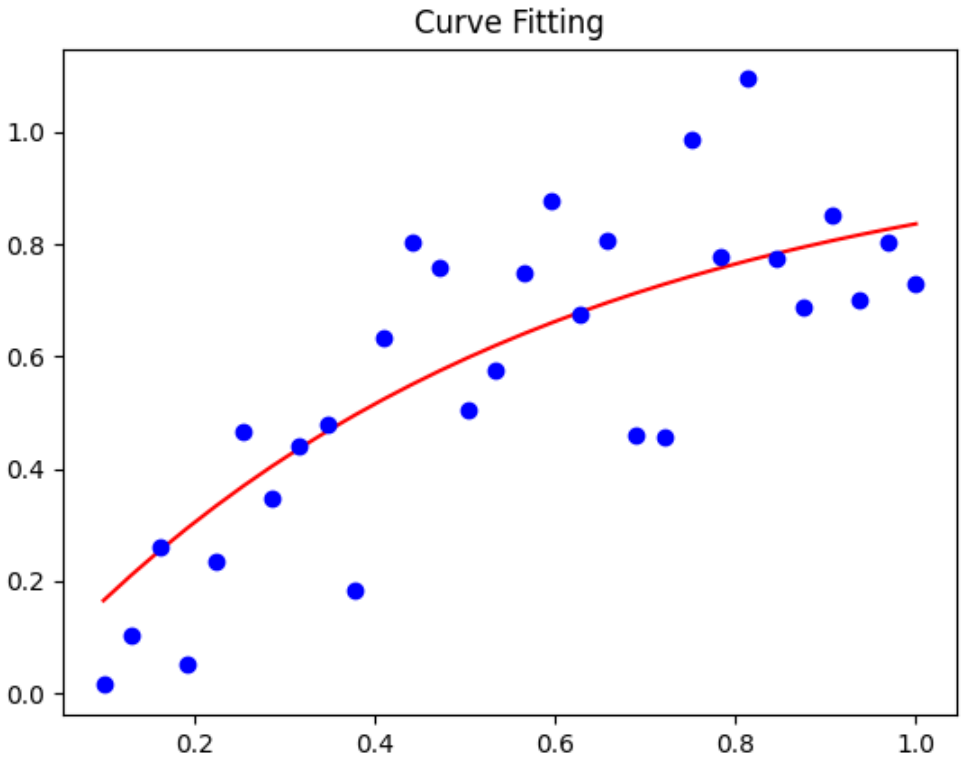
d) When the function surface serves as a concave surface and the artificial object/light point is beyond centre of curvature

En la imagen se demuestra la idea detrás de OIO para generar nuevas soluciones en el espacio de búsqueda unidimensional. En esta figura, se supone que un punto de luz artificial (objeto) en el espacio de búsqueda conjunta y objetivo está frente a la superficie de función (espejo) a una distancia particular del vértice (los valores en el eje X forman el espacio de búsqueda / solución y los valores en el eje $f(x)$ forman el espacio objetivo. El conjunto de todos los puntos en el sistema de coordenadas X $f(x)$ forma la búsqueda conjunta y el espacio objetivo). La imagen artificial se forma en el espacio conjunto de búsqueda y objetivo y su posición y altura se determina mediante ecuaciones de espejo y aumento. Finalmente, mapear la posición de la imagen artificial en el espacio de búsqueda da como resultado la posición de la nueva solución en el espacio de búsqueda. Según el tipo de parte reflectante de la superficie funcional (convexa o cóncava) y según la posición del punto de luz artificial (objeto) en el espacio de búsqueda conjunta y objetivo, existen cuatro situaciones diferentes bajo las cuales se generan nuevas soluciones. La idea detrás de OIO es así simple. Dada una solución individual O en la población, se elige al azar una solución diferente $F(puntodevértice)$ de la población. Si F es peor que O, en términos de función / valor objetivo, entonces se supone que la superficie es convexa y la nueva solución se genera en posición vertical en algún lugar hacia O, en la línea que conecta O y F. Si F es mejor que O, entonces se supone que la superficie es cóncava y la nueva solución se genera en posición vertical hacia o invertida hacia afuera O, en la línea que conecta O y F en la búsqueda espacio.

4. Implementación en Python

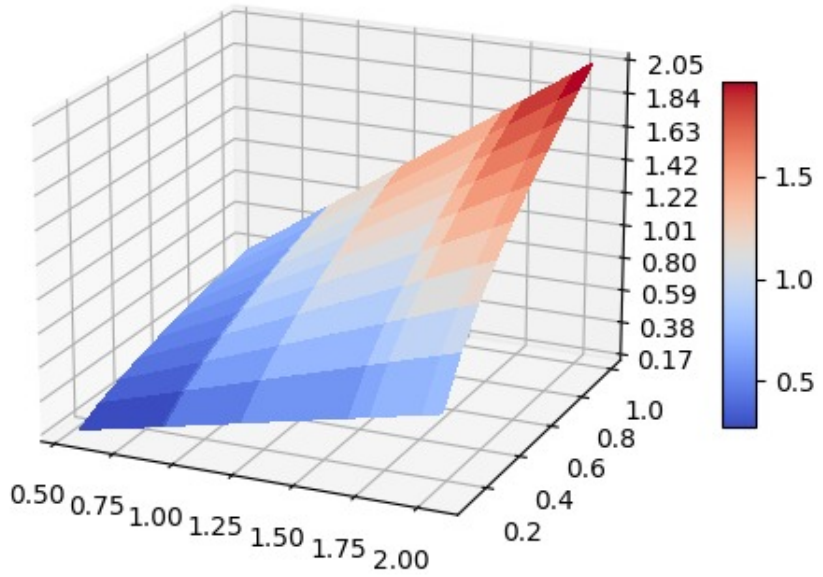
4.1. Curve Fitting

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import *

def Weibull(a, t):
    return 1 - np.exp(-(t) * a)

t = np.linspace(0.1,1.0,30)
a = Weibull(2.0, t) + 0.2 * np.random.normal(size=len(t))
popt, pcov = curve_fit(Weibull, t, a)

plt.title("Curve Fitting")
plt.plot(t, Weibull(*popt, t),'r-')
plt.plot(t,a,"bo")
plt.show()
```



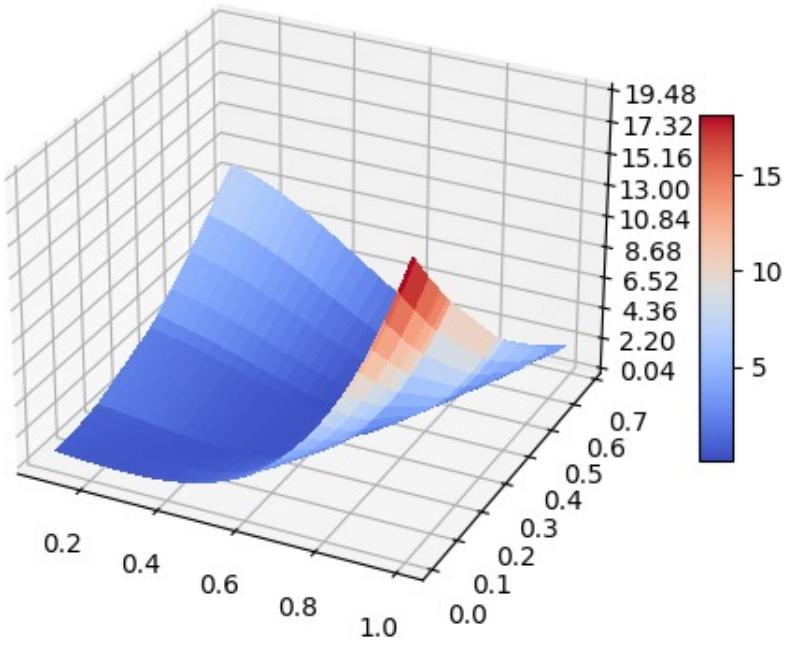
4.2. Higuchi

```
def Higuchi(a, t):
    return a * (t**(1 / 2))
Rosenbrok = lambda x,y: ((1 - x)**2 + 20*(y - x**2)**2)

b = np.array([0.2,0.3,0.45,0.55,0.6,0.7,0.75,0.8,0.8,0.8])
t = np.linspace(0.1,1.0,len(b))
a = Higuchi(2.0, t) + 0.2 * np.random.normal(size=len(b))

popt, pcov = curve_fit(Peppas, a, b)

a,t = np.meshgrid(a,t)
bb = Higuchi(a, t)
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(a, t, bb, cmap=cm.coolwarm)

ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

fig.colorbar(surf, shrink=0.5, aspect=10)
plt.show()
```



4.3. Rosenbrock

```
def Weibull(a, t):
    return 1 - np.exp(-(t) * a)
Rosenbrok = lambda x,y: ((1 - x)**2 + 20*(y - x**2)**2)

t = np.linspace(0.1,1.0,30)
a = Weibull(2.0, t) + 0.2 * np.random.normal(size=len(t))

x = np.linspace(0.1,1.0,30)
y = np.array(Weibull(a, t))
x, y = np.meshgrid(x,y)
z = Rosenbrok(x, y)
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(x,y,z, cmap=cm.coolwarm)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

fig.colorbar(surf, shrink=0.5, aspect=10)
plt.show()
```



5. Conclusión

A nivel general, la optimización puede realizarse en diversos ámbitos, pero siempre con el mismo objetivo: mejorar el funcionamiento de algo o el desarrollo de un proyecto a través de una gestión perfeccionada de los recursos. La optimización puede realizarse en distintos niveles, aunque lo recomendable es concretarla hacia el final de un proceso, significa realizar una tarea de la mejor manera posible.