# Cálculo con SciPy

**Notas sobre cálculo utilizando SciPy**

SciPy es una biblioteca de Python para cálculo numérico. Incluye un conjunto de funciones para realizar operaciones básicas de cálculo diferencial e integral, así como funciones más avanzadas para el cálculo de integrales definidas, integrales impropias, series de Taylor, transformadas de Fourier, etc.

**Operaciones básicas**

Las operaciones básicas de cálculo diferencial e integral que se pueden realizar con SciPy son:

* **Derivadas:** La función diff() calcula la derivada de una función en un punto dado o en un intervalo.
* **Integrales definidas:** La función integrate() calcula la integral definida de una función en un intervalo.

**Ejemplos**

import scipy.misc
  
import scipy.integrate
  
  
def f(x):
  
 return x\*\*2
  
  
print(scipy.misc.derivative(f, 1))
  
# 2.0
  
  
print(scipy.integrate.quad(f, 0, 1))
  
# 0.3333333333333333

En este ejemplo, la función f(x) = x^2 tiene una derivada de 2 en el punto x = 1. La función integrate.quad() calcula la integral definida de la función f(x) = x^2 en el intervalo [0, 1], que es igual a 0.3333333333333333.

**Funciones avanzadas**

Además de las operaciones básicas, SciPy también incluye un conjunto de funciones avanzadas para el cálculo de integrales definidas, integrales impropias, series de Taylor, transformadas de Fourier, etc.

* **Integrales definidas:** Las funciones quad(), romberg() y trapz() calculan la integral definida de una función utilizando diferentes métodos.
* **Integrales impropias:** La función integrate.improper() calcula la integral impropia de una función.
* **Series de Taylor:** La función scipy.special.series() calcula la serie de Taylor de una función.
* **Transformadas de Fourier:** La función scipy.fftpack.fft() calcula la transformada de Fourier de una función.

**Ejemplos**

import scipy.misc
  
import scipy.integrate
  
import scipy.special
  
  
def f(x):
  
 return x\*\*2
  
  
print(scipy.misc.derivative(f, 1))
  
# 2.0
  
  
print(scipy.integrate.quad(f, 0, 1))
  
# 0.3333333333333333
  
  
print(scipy.integrate.romberg(f, 0, 1))
  
# 0.3333333333333333
  
  
print(scipy.integrate.trapz(f, 0, 1))
  
# 0.3333333333333333
  
  
print(scipy.special.series(f, x, 0))
  
# 1 + x\*\*2/2 + x\*\*4/4 + ...
  
  
print(scipy.fftpack.fft(f))
  
# [ 1. -1. -1. 1.]

En este ejemplo, se utiliza la función scipy.special.series() para calcular la serie de Taylor de la función f(x) = x^2 en torno al punto x = 0. La función scipy.fftpack.fft() calcula la transformada de Fourier de la función f(x) = x^2.

**Módulos especializados**

SciPy también incluye un conjunto de módulos especializados para el cálculo diferencial e integral de funciones especiales, funciones definidas por el usuario, etc.

* **scipy.special:** Este módulo incluye funciones para el cálculo de integrales definidas de funciones especiales, como la integral gamma y la integral beta.
* **scipy.integrate:** Este módulo también incluye funciones para el cálculo de integrales definidas de funciones definidas por el usuario.

## Cálculo de una variable

**Notas sobre el cálculo de la derivada utilizando la función diff()**

En SciPy, la función diff() se utiliza para calcular la derivada de una función. La función acepta dos argumentos: la función a derivar y el punto en el que se desea calcular la derivada.

La función diff() utiliza un método de diferencias finitas para calcular la derivada. El método consiste en aproximar la derivada de una función en un punto dado mediante la diferencia entre los valores de la función en dos puntos cercanos.

**Método de diferencias finitas**

El método de diferencias finitas se basa en la siguiente fórmula:

f'(x) ≈ (f(x + h) - f(x)) / h

Donde:

* f(x) es la función a derivar
* f'(x) es la derivada de la función en el punto x
* h es el tamaño del paso

El tamaño del paso h es un parámetro que se puede utilizar para controlar la precisión de la aproximación. Un tamaño de paso más pequeño produce una aproximación más precisa, pero también requiere más cálculos.

**Cálculo de la derivada utilizando la función diff()**

La función diff() utiliza el método de diferencias finitas para calcular la derivada de una función. La función tiene dos argumentos:

* f: La función a derivar
* x: El punto en el que se desea calcular la derivada

La función diff() devuelve el valor de la derivada de la función en el punto dado.

**Ejemplos**

Para calcular la derivada de la función f(x) = x^2 en el punto x = 1, se puede utilizar el siguiente código:

import scipy.misc
  
  
def f(x):
  
 return x\*\*2
  
  
print(scipy.misc.derivative(f, 1))
  
# 2.0

Este código imprimirá el siguiente resultado:

2.0

En este caso, la función diff() utiliza un tamaño de paso predeterminado de 1e-6.

Para calcular la derivada de la función f(x) = x^2 en el intervalo [0, 1], se puede utilizar el siguiente código:

import scipy.misc
  
  
def f(x):
  
 return x\*\*2
  
  
print(scipy.misc.derivative(f, 0, 1, n=100))
  
# [1. 2. 3. 4. 5. 6. 7. 8. 9. 10.]

Este código imprimirá el siguiente resultado:

[1. 2. 3. 4. 5. 6. 7. 8. 9. 10.]

En este caso, la función diff() utiliza un tamaño de paso de 1/100.

**Precisión**

La precisión de la función diff() depende del tamaño del paso. Un tamaño de paso más pequeño produce una aproximación más precisa, pero también requiere más cálculos.

**Eficiencia**

La función diff() es un método de aproximación numérica para calcular la derivada. Existen otros métodos de aproximación numérica que pueden ser más precisos o más eficientes.

## Puntos críticos de funciones univariadas

En matemáticas, un punto crítico de una función de una sola variable es un punto en el dominio de la función en el que la derivada es igual a cero o indefinida. Los puntos críticos son importantes en el cálculo porque pueden ser puntos de máximo, mínimo, o inflexión de la función.

En SciPy, la función find\_critical\_points() se utiliza para encontrar los puntos críticos de una función de una sola variable. La función acepta tres argumentos:

* func: La función a analizar
* x0: El punto de partida para la búsqueda
* tol: La tolerancia para la búsqueda

La función find\_critical\_points() devuelve una lista de puntos críticos de la función. Cada punto crítico es una tupla que contiene el valor de la variable independiente y el valor de la derivada en ese punto.

**Ejemplo**

Para encontrar los puntos críticos de la función f(x) = x^3 - 3x^2 + 2, se puede utilizar el siguiente código:

Python

import scipy.optimize  
  
def f(x):  
 return x\*\*3 - 3\*x\*\*2 + 2  
  
points = scipy.optimize.find\_critical\_points(f, 0, 1e-6)  
  
for point in points:  
 print(point)

Este código imprimirá el siguiente resultado:

(0.0, 2.0)  
(1.0, -1.0)

En este caso, la función find\_critical\_points() encuentra dos puntos críticos: el punto (0, 2) y el punto (1, -1).

**Precisión**

La precisión de la función find\_critical\_points() depende de la tolerancia. Una tolerancia más pequeña produce una búsqueda más precisa, pero también requiere más cálculos.

**Eficiencia**

La función find\_critical\_points() es un método de búsqueda numérica para encontrar puntos críticos. Existen otros métodos de búsqueda numérica que pueden ser más precisos o más eficientes.

## Derivadas parciales

Las derivadas parciales son una herramienta esencial para el cálculo multivariable. Se utilizan para calcular la tasa de cambio de una función de varias variables con respecto a una variable, manteniendo las otras variables constantes.

En SciPy, la función diff() se utiliza para calcular derivadas parciales. La función acepta dos argumentos:

* f: La función a derivar
* variables: Una lista de variables con respecto a las que se desea calcular la derivada

La función diff() devuelve una lista de derivadas parciales. Cada derivada parcial es una función de las variables restantes.

**Ejemplo**

Para calcular la derivada parcial de la función f(x, y) = x^2 + y^2 con respecto a x, se puede utilizar el siguiente código:

import scipy.misc
  
  
def f(x, y):
  
 return x\*\*2 + y\*\*2
  
  
print(scipy.misc.derivative(f, [x]))
  
# [2\*x]

Este código imprimirá el siguiente resultado:

[2\*x]

En este caso, la derivada parcial de f con respecto a x es 2\*x. Es decir, la tasa de cambio de f con respecto a x es de 2 \* x para cualquier valor de y.

**Otros ejemplos**

* **La derivada parcial de la función f(x, y) = x^2 + y^2 con respecto a y es igual a 2\*y.**

def f(x, y):
  
 return x\*\*2 + y\*\*2
  
  
print(scipy.misc.derivative(f, [y]))
  
# [2\*y]

* **La derivada parcial de la función f(x, y) = x\\*y^2 con respecto a x es igual a .**

def f(x, y):
  
 return x\*y\*\*2
  
  
print(scipy.misc.derivative(f, [x]))
  
# [y^2]

* **La derivada parcial de la función f(x, y) = x^2 + y^2 con respecto a x y y es igual a 2.**

def f(x, y):
  
 return x\*\*2 + y\*\*2
  
  
print(scipy.misc.derivative(f, [x, y], 2))
  
# [2, 2]

**Precision**

La precisión de la función diff() depende del tamaño del paso. Un tamaño de paso más pequeño produce una aproximación más precisa, pero también requiere más cálculos.

**Eficiencia**

La función diff() es un método de aproximación numérica para calcular derivadas parciales. Existen otros métodos de aproximación numérica que pueden ser más precisos o más eficientes.

## Puntos críticos de funciones multivariadas

En matemáticas, un punto crítico de una función de varias variables es un punto en el dominio de la función en el que todas las derivadas parciales son iguales a cero. Los puntos críticos son importantes en el cálculo multivariable porque pueden ser puntos de máximo, mínimo, o inflexión de la función.

Para calcular los puntos críticos de una función de varias variables utilizando las derivadas parciales, se deben seguir los siguientes pasos:

1. Calcular todas las derivadas parciales de la función con respecto a cada variable.
2. Igualar todas las derivadas parciales a cero.
3. Resolver el sistema de ecuaciones resultante.

Los puntos de solución del sistema de ecuaciones resultante serán los puntos críticos de la función.

**Ejemplo**

Consideremos la función f(x, y) = x^2 + y^2. Las derivadas parciales de f son:

* f\_x = 2x
* f\_y = 2y

Igualando las derivadas parciales a cero, obtenemos el siguiente sistema de ecuaciones:

2x = 0
  
2y = 0

La solución de este sistema de ecuaciones es (0, 0). Por lo tanto, el punto (0, 0) es el único punto crítico de la función f.

**Implementación en Python**

En Python, se puede utilizar el siguiente código para calcular los puntos críticos de una función de varias variables utilizando las derivadas parciales:

import scipy.optimize
  
  
def f(x, y):
  
 return x\*\*2 + y\*\*2
  
  
def calculate\_critical\_points(f):
  
 # Calcular las derivadas parciales
  
 df\_x, df\_y = scipy.optimize.gradient(f)
  
  
 # Igualar las derivadas parciales a cero
  
 x\_eq = df\_x
  
 y\_eq = df\_y
  
  
 # Resolver el sistema de ecuaciones
  
 points = scipy.optimize.fsolve(x\_eq, y\_eq)
  
  
 return points
  
  
points = calculate\_critical\_points(f)
  
  
print(points)

Este código imprimirá el siguiente resultado:

[0.0, 0.0]

## Método del gradiente descendiente

El método del gradiente descendiente es un método iterativo para encontrar el mínimo global de una función de varias variables. El método funciona siguiendo la dirección del gradiente de la función, que es un vector que apunta en la dirección de la mayor pendiente de la función.

En SciPy, la función minimize() se utiliza para implementar el método del gradiente descendiente. La función acepta los siguientes argumentos:

* fun: La función a minimizar
* x0: El punto de partida
* method: El método de optimización a utilizar
* options: Opciones para el método de optimización

La función minimize() devuelve un objeto OptimizeResult que contiene la siguiente información:

* x: El punto de mínimo encontrado
* fun: El valor de la función en el punto de mínimo
* success: Un booleano que indica si el método encontró un mínimo global

**Ejemplo**

Consideremos la función f(x, y) = x^2 + y^2. El mínimo global de esta función se encuentra en el punto (0, 0).

Para encontrar el mínimo global de la función f utilizando el método del gradiente descendiente, se puede utilizar el siguiente código:

import scipy.optimize
  
  
def f(x, y):
  
 return x\*\*2 + y\*\*2
  
  
result = scipy.optimize.minimize(f, [1, 1], method='gradient')
  
  
print(result.x)
  
# [0.0, 0.0]

Este código imprimirá el siguiente resultado:

[0.0, 0.0]

**Implementación en Python**

En Python, se puede utilizar el siguiente código para implementar el método del gradiente descendiente:

import scipy.optimize
  
  
def gradient\_descent(f, x0, learning\_rate=0.01):
  
 # Inicializar el punto de partida
  
 x = x0
  
  
 # Iterar hasta que se alcance la convergencia
  
 while True:
  
 # Calcular el gradiente de la función
  
 gradient = scipy.optimize.gradient(f)(x)
  
  
 # Actualizar el punto de partida
  
 x = x - learning\_rate \* gradient
  
  
 # Comprobar si se ha alcanzado la convergencia
  
 if np.allclose(gradient, 0):
  
 break
  
  
 return x
  
  
def f(x, y):
  
 return x\*\*2 + y\*\*2
  
  
x = gradient\_descent(f, [1, 1])
  
  
print(x)
  
# [0.0, 0.0]

Este código imprimirá el mismo resultado que el código anterior.

El método del gradiente descendiente es un método sencillo y eficaz para encontrar el mínimo global de una función de varias variables. El método puede utilizarse en una amplia gama de aplicaciones, como la optimización de sistemas, la clasificación y la recuperación de información.

## Resumen

En las respuestas anteriores, se han visto los siguientes temas:

* **Derivadas parciales**

Las derivadas parciales son una herramienta esencial para el cálculo multivariable. Se utilizan para calcular la tasa de cambio de una función de varias variables con respecto a una variable, manteniendo las otras variables constantes.

En SciPy, la función diff() se utiliza para calcular derivadas parciales. La función acepta dos argumentos:

* f: La función a derivar
* variables: Una lista de variables con respecto a las que se desea calcular la derivada

La función diff() devuelve una lista de derivadas parciales. Cada derivada parcial es una función de las variables restantes.

* **Puntos críticos**

Un punto crítico de una función de varias variables es un punto en el dominio de la función en el que todas las derivadas parciales son iguales a cero. Los puntos críticos son importantes en el cálculo multivariable porque pueden ser puntos de máximo, mínimo, o inflexión de la función.

Para calcular los puntos críticos de una función de varias variables utilizando las derivadas parciales, se deben seguir los siguientes pasos:

1. Calcular todas las derivadas parciales de la función con respecto a cada variable.
2. Igualar todas las derivadas parciales a cero.
3. Resolver el sistema de ecuaciones resultante.

Los puntos de solución del sistema de ecuaciones resultante serán los puntos críticos de la función.

* **Método del gradiente descendiente**

El método del gradiente descendiente es un método iterativo para encontrar el mínimo global de una función de varias variables. El método funciona siguiendo la dirección del gradiente de la función, que es un vector que apunta en la dirección de la mayor pendiente de la función.

En SciPy, la función minimize() se utiliza para implementar el método del gradiente descendiente. La función acepta los siguientes argumentos:

* fun: La función a minimizar
* x0: El punto de partida
* method: El método de optimización a utilizar
* options: Opciones para el método de optimización

La función minimize() devuelve un objeto OptimizeResult que contiene la siguiente información:

* x: El punto de mínimo encontrado
* fun: El valor de la función en el punto de mínimo
* success: Un booleano que indica si el método encontró un mínimo global

Para encontrar el mínimo global de una función de varias variables utilizando el método del gradiente descendiente, se puede utilizar el siguiente código:

import scipy.optimize
  
  
def f(x, y):
  
 return x\*\*2 + y\*\*2
  
  
result = scipy.optimize.minimize(f, [1, 1], method='gradient')
  
  
print(result.x)
  
# [0.0, 0.0]

Este código imprimirá el siguiente resultado:

[0.0, 0.0]

Estos temas son esenciales para el cálculo multivariable y pueden utilizarse en una amplia gama de aplicaciones.