COLOMBIA POTENCIA DE LA VIDA

TIC

Actividad 1

Introducción a La convolución

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Introducción a La convolución

La convolución es una operación matemática fundamental en el procesamiento de señales y el procesamiento de imágenes.

En el procesamiento de señales

La convolución es una operación fundamental que se utiliza para aplicar filtros a una señal con el fin de resaltar ciertas características o realizar operaciones específicas. La convolución se utiliza comúnmente en aplicaciones como el filtrado de señales, la detección de características, la equalización de histogramas, entre otros.

La convolución en el procesamiento de señales implica multiplicar dos funciones, la señal de entrada y el kernel del filtro, y luego integrar el producto resultante en un intervalo específico. El kernel del filtro, también conocido como función de respuesta al impulso, especifica cómo se debe modificar la señal de entrada.

En el procesamiento de señales

Para comprender mejor el concepto, consideremos un ejemplo simple de convolución en el procesamiento de señales. Supongamos que tenemos una señal de entrada x(t) y un filtro con respuesta al impulso h(t). La convolución de la señal de entrada con el filtro se denota como y(t)=x(t)□h(t) y se calcula de la siguiente manera:

$$y(t) = \int_{-\infty}^{\infty} x(\tau) \cdot h(t - \tau)\, d\tau$$

Donde:

- **x(t)** es la señal de entrada.
- **h(t)** es la respuesta al impulso del filtro.
- **y(t)** es la señal de salida resultante.
- **τ** es una variable de integración.

En el procesamiento de señales

En este ejemplo, la convolución de la señal de entrada **x(t)** con el filtro **h(t)** produce una nueva señal **y(t)**) que resalta ciertas características de interés en la señal original, como la eliminación de ruido, la suavización de la señal o la detección de bordes, dependiendo del diseño del filtro.

Por ejemplo, si **h(t)** es un filtro paso bajo, la convolución con **x(t)** puede atenuar las frecuencias altas en la señal de entrada, lo que resulta en una versión suavizada de la señal. Del mismo modo, si **h(t)** es un filtro paso alto, la convolución puede resaltar las frecuencias altas en la señal de entrada, lo que resulta en la detección de bordes o características de alta frecuencia.

La convolución en el procesamiento de señales es una operación esencial que se utiliza para aplicar filtros y realizar diversas operaciones en las señales, lo que permite resaltar, suavizar o modificar ciertas características de interés.

Ejemplo numérico de convolución discreta en una dimensión (1D):

Supongamos que tenemos dos señales discretas de entrada x[n] y h[n], donde n representa el índice discreto:

$$x[n] = [2, 1, 0, 1, 2]$$
$$h[n] = [1, -1]$$

Queremos convolucionar la señal x[n] con el filtro h[n] para obtener la señal de salida y[n]. La convolución discreta se realiza multiplicando y sumando los elementos de x[n] y h[n] para cada desplazamiento del índice nn.

La fórmula de convolución discreta es:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

Ejemplo numérico de convolución discreta en una dimensión (1D):

Pero en la práctica, para señales discretas de longitud finita, solo necesitamos considerar las muestras disponibles.

Para calcular y[n], desplazamos h[n] a lo largo de x[n] y multiplicamos elemento por elemento, y luego sumamos los productos resultantes.

Aplicando esto a nuestro ejemplo:

Ponemos h[n] al inicio y multiplicamos con x[n]:

Para n=0:

y[0]=(2□1)+(1□-1)+(0□0)+(1□0)+(2□0)=1

Desplazamos h[n] una posición a la derecha:

Para n=1n=1:

y[1]=(2□0)+(1□1)+(0□-1)+(1□0)+(2□0)=1

Continuamos el proceso hasta que hemos desplazado completamente h[n] a través de x[n].

Entonces, la señal de salida sería:

y[n]=[-1,-1,1,1]

Este es el resultado de convolucionar la señal de entrada x[n] con el filtro h[n] utilizando convolución discreta en una dimensión.

Para realizar una convolución 1D en Python

Puedes utilizar la función numpy.convolve(). ejemplo

```
import numpy as np

# Definir las señales de entrada y el filtro
x = np.array([2, 1, 0, 1, 2])  # Señal de entrada
h = np.array([1, -1])          # Filtro

# Realizar la convolución
y = np.convolve(x, h, mode='valid')

# Imprimir el resultado
print("Señal de salida después de la convolución:")
print(y)
```

```
Señal de salida después de la convolución:
[-1 -1  1  1]
```

Para realizar una convolución 1D en Python

ver la documentación oficial de np.convolve en:

https://numpy.org/doc/stable/reference/generated/numpy.convolve.html

En este ejemplo, x es la señal de entrada y h es el filtro. Luego, utilizamos np.convolve(x, h, mode='valid') para realizar la convolución. El argumento mode='valid' indica que solo queremos los valores de la convolución donde las señales se superponen completamente, es decir, sin relleno.
El resultado se almacena en y, que es la señal de salida después de la convolución. Puedes imprimir y para ver la señal resultante.

En el contexto del procesamiento de imágenes

La convolución se utiliza para aplicar filtros o máscaras a una imagen para resaltar ciertas características o realizar operaciones específicas, como el suavizado, el realce de bordes o la detección de características.

En términos generales, la convolución implica superponer una ventana de tamaño específico, llamada kernel o máscara, en cada píxel de la imagen y realizar una operación de suma ponderada. El resultado de esta operación se asigna al píxel central de la ventana y genera una nueva imagen convolucionada.

En el contexto del procesamiento de imágenes

Para entender mejor el proceso de convolución, consideremos una imagen de entrada I y un kernel K. La operación de convolución se realiza calculando la suma ponderada de los elementos de I bajo la ventana del kernel K. La fórmula general para la convolución de una imagen I con un kernel K en un punto (x,y) se puede expresar como:

$$I'(x,y) = \sum_i \sum_j I(x+i, y+j) \cdot K(i,j)$$

Donde:

I'(x,y) es el valor resultante en el punto (x,y) de la imagen convolucionada.
I(x+i,y+j) son los valores de los píxeles vecinos de I bajo la ventana del kernel.
K(i,j) son los valores del kernel.

En el contexto del procesamiento de imágenes

El proceso de convolución puede llevarse a cabo de diferentes maneras dependiendo del tipo de borde de la imagen, el tratamiento de los píxeles en los bordes y la normalización de los valores resultantes. Además, se pueden utilizar diferentes kernels para lograr diferentes efectos en la imagen, como suavizado, realce de bordes, detección de características, entre otros.

La convolución es una operación fundamental en el procesamiento de imágenes que permite aplicar filtros y realizar diversas operaciones para mejorar, resaltar o extraer características de una imagen. Es ampliamente utilizada en aplicaciones de visión por computadora, procesamiento de imágenes médicas, análisis de imágenes satelitales y muchas otras áreas.

Ejemplo numérico de convolución 2D discreto

Supongamos que tenemos la siguiente matriz de entrada y el filtro:

Matriz de entrada:

[1, 2, 3, 3],
[4, 5, 6, 6],
[7, 8, 9, 9],
[7, 8, 9, 9]

Filtro:

[[0, 1, 0],
[-1, 1, -1],
[0, 1, 0]]

Para realizar la convolución 2D, colocamos el filtro sobre la matriz de entrada y multiplicamos elemento por elemento, luego sumamos los resultados. Este proceso se repite para cada ubicación posible del filtro en la matriz de entrada.

Ejemplo numérico de convolución 2D discreto

```python
import numpy as np
from scipy.signal import convolve2d

# Definir la matriz de entrada (señal) y el kernel
input_matrix = np.array([[1, 2, 3, 3],
                         [4, 5, 6, 6],
                         [7, 8, 9, 9],
                         [7, 8, 9, 9]])

kernel = np.array([[0,  1,  0],
                   [-1, 1, -1],
                   [0,  1,  0]])

# Realizar la convolución 2D
output_matrix = convolve2d(input_matrix, kernel, mode='valid')

print("Matriz de salida después de la convolución:")
print(output_matrix)
```

```
Matriz de salida después de la convolución:
[[5 7]
 [5 7]]
```

El resultado de la convolución sería una nueva matriz, que representa la señal de salida después de aplicar el filtro.

[[5 7]
 [5 7]]

En Python, puedes realizar una convolución 2D utilizando la biblioteca NumPy. Supongamos que tienes una matriz de entrada input_matrix y un kernel kernel. Puedes usar la función convolve2d de NumPy para realizar la convolución 2D de la siguiente manera:

Ejemplo numérico de convolución 2D discreto

El argumento mode='valid' indica que solo se realizará la convolución en regiones donde el kernel y la entrada coincidan completamente, lo que resulta en una matriz de salida más pequeña.

El resultado será la matriz de salida después de aplicar la convolución 2D con el kernel especificado a la matriz de entrada.

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira