

Práctica 4

Almeraya, Kimberly., Tinoco, Sergio., De Los Ríos, Flavio.
Instituto Politécnico Nacional, Escuela Superior de Cómputo.

Procesamiento de Señales

10 de diciembre de 2022

En esta práctica, se realiza la convolución y correlación de señales utilizando las herramientas que nos proporciona Python 3.10 como la creación de arreglos con ceros, matrices nulas, producto punto, suma de vectores, así como realizar la visualización de estas señales en un gráfico de valores discretos.

Palabras clave: señales, convolución, correlación

aprendizaje automático, análisis de datos y educación.

I. INTRODUCCIÓN

La convolución de señales permite predecir el comportamiento de una señal dada una entrada en particular es por esto que resulta en una operación fundamental tanto en análisis como en el procesamiento de señales. De igual forma la correlación de señales nos determina el grado de semejanza de 2 señales.

La práctica se realizó con la finalidad de ver cómo van cambiando las señales a través de la convolución usando el lenguaje Python, desarrollándose en el entorno de Google Colab. Python permite hacer la convolución multiplicando arreglo por arreglo, señal por señal dando como resultado su gráfico.

Google Colab permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador. Es especialmente adecuado para tareas de

Python es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina.

Recordando que una señal es un tipo de señal en que cada signo que codifica el contenido de la misma puede ser analizado en término de algunas magnitudes que representan valores discretos, en lugar de valores dentro de un cierto rango.

A una señal se le pueden aplicar distintos procesos, como la convolución, que permite conocer qué le va a pasar a una

señal después de “pasar” por un determinado dispositivo u objeto.

En matemáticas y en particular análisis funcional, una convolución es cuando un operador matemático transforma dos funciones ‘ $f(x)$ ’ y ‘ $g(x)$ ’ en una tercera función (resultante) que en cierto sentido representa la magnitud en la que se superponen $f(x)$ y una versión trasladada e invertida de $g(x)$. Una convolución es un tipo muy general de medida móvil, como se puede observar una de las funciones se toma como la función característica de un intervalo.

Con la ayuda de Google Coolab y sus librerías como Numpy y Matplotlib, se puede calcular el resultado de una convolución de dos señales distintas.

OBJETIVO Y PLANTEAMIENTO DEL PROBLEMA

El objetivo de esta práctica fue visualizar con mayor detalle cómo se implementa la convolución y correlación de dos señales distintas así como entender el comportamiento de la señal resultante en función de sus entradas.

Ya que este tipo de operaciones suelen ser algo confusas en cuanto a su comportamiento y con una herramienta tan versátil como Python se puede entender mejor este comportamiento.

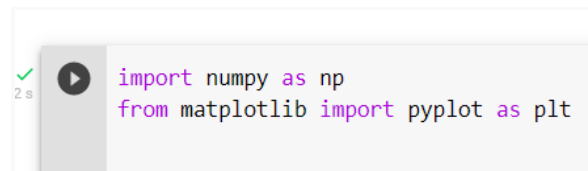
II. DESARROLLO EXPERIMENTAL

Para comenzar el desarrollo lo primero fue crear un nuevo cuaderno en Google

Coolab, el cual nos permite programar rápidamente en Python, de una manera fácil y sencilla.

Para dicha convolución y graficación de esta señal, se fue jugando con el orden en el que se hacía la convolución (pivote) para ir recorriendo y multiplicando ambos arreglos unidimensionales.

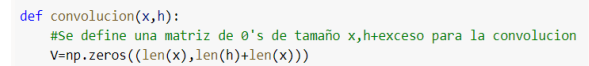
primero importamos las librerías Numpy y Pylot de Matplotlib tal como se muestra en la siguiente imagen:



```
import numpy as np
from matplotlib import pyplot as plt
```

Figura 1. Importación de librerías en python

Luego se crea una matriz para la convolución:



```
def convolucion(x,h):
    #Se define una matriz de 0's de tamaño x,h+exceso para la convolucion
    V=np.zeros((len(x),len(h)+len(x)))
```

Figura 2. Creación de una matriz en python

Se crea un ciclo for para hacer la convolución (multiplicación de señales) y se guarda en la matriz creada:



```
#Se itera en un for anidado
for i in range(len(x)):
    for j in range(len(h)):
        V[i][j+i]=(h[j]*x[i])    #Producto de cada valor de x con cada valor de h
```

Figura 3. Iteración para el llenado de la matriz

Retornamos la suma por columnas de la matriz resultante:

```
return sum(V)
```

Figura 5. Retorno

Se define el arreglo de la señal de salida:

```
def correlacion(x,y,n):
    r=list(np.zeros(n))
```

Figura 6. Arreglo

Añadimos los ceros a los extremos de la señal Y:

```
for i in range(k):
    y.insert(0,0)
    y.insert(len(y),0)
```

Figura 7. Iteración de llenado con 0's

Se aplica un producto punto para la suma de productos, de tal forma que cada producto resulte en un valor de r:

```
for l in range(-k,k+1): #Se itera de -k a k pasando por 0
    r[k+l]=np.dot(x,y[k-l:(k-l)+len(x)]) #Producto punto de la señal desplazada l unidades
return r
```

Figura 8. Iteración de un producto punto

Se define el intervalo de n:

```
nx= np.arange(-3, 4, 1)
```

Figura 9. Intervalo

Se declara la señal x1 como una función particionada:

```
x1=np.pieceswise(nx, [nx == 0, nx ==1,nx==2,nx<0,nx>2], [2,3,2,0,0])
```

Figura 10. Función particionada

Definimos el intervalo de n para la otra señal:

```
nh=np.arange(-2, 3, 1)
```

Figura 11. Intervalo

Se declara h como otra función particionada:

```
h=np.pieceswise(nh, [nh == 1, nh ==-1,nh==0,nh<-1,nh>1], [1,1,2,0,0])
```

Figura 12. Función particionada

Aplicamos la convolución a la primera señal:

```
y=convolucion(x1,h)
print("Señal x[n]:",x1)
print("Señal h[n]:",h)
print("Suma de convolucion:",y)
```

Figura 13. Convolución de X1

Se procede a plotear las señales:

```
plt.figure(figsize=(12,4))
plt.suptitle("Convolucion")
plt.subplot(1,3,1),plt.stem(nx,x1),plt.title("Señal x[n]")
plt.subplot(1,3,2),plt.stem(nh,h),plt.title("Señal h[n]")
plt.subplot(1,3,3),plt.stem(np.arange(-(len(y)//2), (len(y)//2), 1),y),plt.title("Señal y[n]")
plt.show()
```

Figura 14. Ploteo de Señales

Para la correlación ponemos un tamaño de la muestra, siempre cuidando que el valor que elijamos sea un impar, ya que python siempre cuenta a partir del cero:

```
n=9
x2=[2,-1,3,7,1,2,-3]
y=[1,-1,2,-2,4,1,-2,5]
```

Figura 15. Tamaño de la muestra, señal X1 y X2

Se aplica la correlación a la nueva señal X:

```
r=correlacion(x2,y,n)

print("Señal x[n]:",x2)
print("Señal y[n]:",y)
print("Correlacion rxy",r)
```

Figura 16. Correlación de X2

Se definen los intervalos para graficar cada señal:

```
nx=np.arange(-(len(x2)//2),(len(x2)//2)+1, 1)
ny=np.arange(-(len(y)//2),(len(y)//2), 1)
nr=np.arange(-(n//2),(n//2)+1, 1)
```

Figura 17. Intervalos

III. DISCUSIÓN Y RESULTADOS

Se procede a plotear las señales y obtenemos:

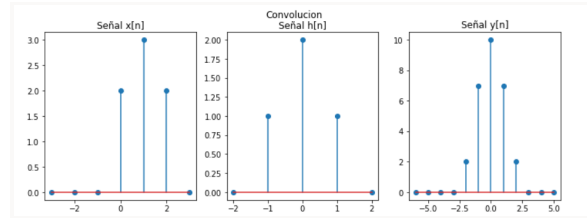


Figura 18. Gráfica convolución

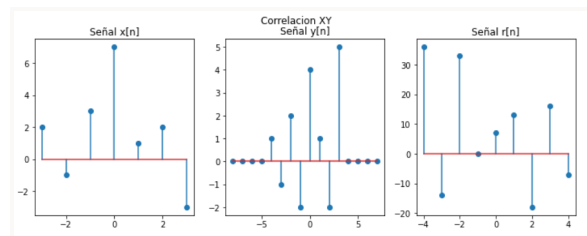


Figura 19. Gráfica correlación XY

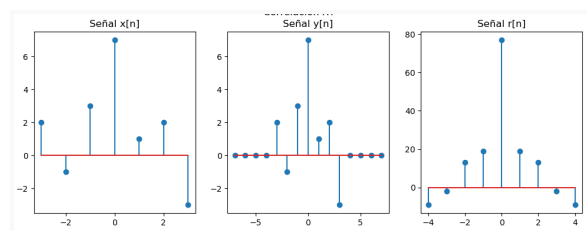


Figura 20. Gráfica autocorrelación XX

IV. CONCLUSIÓN

Para concluir, esta práctica fue de ayuda para la visualización de la convolución y correlación de una señal. Así como para visualizar las transformaciones que sufrirá después de pasar a través de un objeto o dispositivo, por ejemplo, una señal de audio puede sufrir cambios cuando es transmitida a través de un micrófono, podremos saber entonces el ruido detrás de la señal de audio.

La convolución de señales está presente todo el tiempo en la vida cotidiana, por esta razón es importante conocerlas, sobre todo en el campo de la innovación, para futuras aplicaciones.

Las convoluciones de las señales podrán funcionar para la implementación de redes neuronales. Las redes neuronales convolucionales son muy utilizadas para detectar patrones, es decir los patrones son las convoluciones reconocidas, esto quiere decir que podremos detectar ruidos o anomalías dentro de las señales para generar nuevos conocimientos sobre algún tema en específico, ya sean señales de audio y estar alertas en algún ecosistema que se encuentre en peligro, por decir algún ejemplo.

V. REFERENCIAS

Correlación de Señales en TD. (2022, 25 marzo). FCEIA. Recuperado 12 de diciembre de 2022, de https://www.fceia.unr.edu.ar/tesys/html/Correlacion_en_TD_2up.pdf

ENGINEERING THE WORLD FROM PARAGUAY. (2013, 26 diciembre).

Convolución, procesamiento de señales.

Rama Estudiantil del IEEE de la UCSA.

<https://ramaucsa.wordpress.com/2013/12/17/convolucion-procesamiento-de-senales/>