Tutorial Fourier

Atuel Villegas
Blas de Haro Barbás

2023

1 Introducción a la Transformada de Fourier

La Transformada de Fourier es una herramienta matemática que se utiliza para descomponer una señal en sus componentes de frecuencia. Es ampliamente utilizada en diversas áreas, como procesamiento de señales, análisis de datos y procesamiento de imágenes. En este tutorial, aprenderás cómo aplicar la Transformada de Fourier en Python para analizar un conjunto de datos y comprender sus componentes de frecuencia.

2 Preparación de Datos y Bibliotecas

Antes de comenzar, asegúrate de tener Python instalado en tu sistema. Utilizaremos las siguientes bibliotecas:

- **numpy**: para operaciones numéricas y manejo de matrices.
- **matplotlib**: para trazar gráficos y visualizaciones.
- **scipy** para funciones de procesamiento de señales, incluida la Transformada de Fourier.

Asegúrate de instalar estas bibliotecas usando el comando pip install numpy matplotlib scipy.

3 Aplicación de la Transformada de Fourier

Paso 1: Cargar los Datos

Comienza cargando tus datos en forma de una serie temporal. Supongamos que tienes un archivo CSV llamado "datos.csv" con una columna de valores a analizar.

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

# Cargar datos desde el archivo CSV
data = np.loadtxt("datos.csv")
```

Paso 2: Calcular la Transformada de Fourier

Usa la función **scipy.fft.fft** para calcular la Transformada de Fourier. Esto te dará la magnitud y la fase de las componentes de frecuencia.

```
from scipy.fft import fft, fftfreq

# Calcular la Transformada de Fourier
transformed_data = fft(data)
frequencies = fftfreq(len(data))  # Calcular las frecuencias
   correspondientes
```

‘

Paso 3: Visualizar los Resultados

Traza el espectro de frecuencia para visualizar las componentes de frecuencia en tus datos.

```
plt.figure(figsize=(10, 6))
plt.plot(frequencies, np.abs(transformed_data))
plt.title("Espectro de Frecuencia")
plt.xlabel("Frecuencia (Hz)")
plt.ylabel("Amplitud")
plt.grid()
plt.show()
```

4 Interpretación y Conclusiones

En el gráfico resultante, las frecuencias en el eje x representan las componentes de frecuencia presentes en tus datos, mientras que las amplitudes en el eje y indican la importancia relativa de cada frecuencia. Puedes interpretar este gráfico para identificar las frecuencias dominantes en tus datos y extraer información sobre los patrones presentes en la señal. La Transformada de Fourier es una poderosa herramienta para analizar el contenido de frecuencia de un conjunto de datos. Este tutorial te ha guiado a través de los pasos esenciales para aplicar la Transformada de Fourier en Python, cargar los datos, calcular la transformada y visualizar el espectro de frecuencia. Ahora tienes las herramientas para explorar y comprender las características de frecuencia en tus propios conjuntos de datos.

Aquí va un ejemplo completo con la generación de datos inventados para hacer el análisis de la transformada de Fourier:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq


# Generar datos de ejemplo: una senial con dos componentes de
    frecuencia
t = np.linspace(0, 2*np.pi, 1000)  # Tiempo
frequency1 = 5  # Frecuencia en Hz
frequency2 = 20  # Frecuencia en Hz
amplitude1 = 2.0
amplitude2 = 1.0
signal = amplitude1 * np.sin(2*np.pi*frequency1*t) +
   amplitude2 * np.sin(2*np.pi*frequency2*t)


# Calcular la Transformada de Fourier
transformed_data = fft(signal)
frequencies = fftfreq(len(signal), t[1] - t[0])  # Calcular
   las frecuencias correspondientes


# Visualizar los resultados
plt.figure(figsize=(10, 6))
plt.plot(frequencies, np.abs(transformed_data))
plt.title("Espectro de Frecuencia")
plt.xlabel("Frecuencia (Hz)")
```

```
plt.ylabel("Amplitud")
plt.grid()
plt.show()
```

Paso 4: Calcular el Periodograma

El periodograma es una herramienta que se utiliza para estimar la densidad espectral de potencia de una señal. Muestra cómo se distribuye la potencia en función de la frecuencia. En Python, podemos utilizar la función scipy.signal.periodogram para calcular el periodograma de la señal.

```
from scipy.signal import periodogram

# Calcular el periodograma
frequencies_p, power_spectrum = periodogram(signal, fs=1/(t
    [1] - t[0]))
```

Paso 5: Visualizar el Periodograma

```
plt.figure(figsize=(10, 6))

# Espectro de Frecuencia
plt.subplot(2, 1, 1)
plt.plot(frequencies, np.abs(transformed_data))
plt.title("Espectro de Frecuencia")
plt.xlabel("Frecuencia (Hz)")
plt.ylabel("Amplitud")
plt.grid()

# Periodograma
plt.subplot(2, 1, 2)
plt.plot(frequencies_p, power_spectrum)
plt.title("Periodograma")
plt.xlabel("Frecuencia (Hz)")
plt.ylabel("Densidad Espectral de Potencia")
plt.grid()

plt.tight_layout()
plt.show()
```

5 Conclusiones

En la primera parte del gráfico, puedes observar el espectro de frecuencia, que representa las amplitudes de las componentes de frecuencia en la señal. En la segunda parte del gráfico, el periodograma muestra la distribución de la densidad espectral de potencia en función de la frecuencia. El periodograma es especialmente útil para identificar los períodos dominantes en una señal, lo que puede proporcionar información valiosa sobre los patrones temporales en los datos.

6 Resumen

En este tutorial, has aprendido cómo aplicar la Transformada de Fourier y el periodograma en Python para analizar un conjunto de datos y comprender sus componentes de frecuencia y períodos involucrados. Estas herramientas son esenciales para descomponer señales complejas y extraer información importante sobre su contenido en frecuencia y en el dominio del tiempo. Ahora tienes las herramientas para explorar y comprender las características de frecuencia y períodos en tus propios conjuntos de datos.