Отчет по лабораторному заданию №4

ИУ1И-41М Цзинь Сюаньфэн

Апрель 2025

1 Цель задания

Изучить характеристики сигналов мозговых волн по данным ЭЭГ и освоить расчет спектральной плотности и вейвлет-преобразования.

研究脑电图数据中脑电波信号的特征，掌握频谱密度和小波变换的计算方法。

2 Результаты и Обсуждения

2.1 Анализ диаграммы ээг-сигнала

Данные были получены из набора данных Zenodo, содержащего записи ЭЭГ новорожденных с аннотациями приступов. Файлы в наборе данных определяются следующим образом:

ЭЭГ (1 - 79): файл EDF, содержащий записи ЭЭГ с частотой дискретизации 256 Гц. Единицы измерения ЭЭГ - микровольты.

Аннотации_2017: CSV-файл или MAT-файл, содержащий аннотации 3 экспертов для всех 79 новорожденных, сэмплированных с частотой 1 Гц в секунду.

clinical_information: CSV-файл, содержащий клинические данные, полученные из записей пациента, и общую информацию о пространственном распределении последовательных припадков в записях, согласованных с файлом EDF (отмечены только рецензентом A).

数据来自Zenodo的带有癫痫发作注释的新生儿脑电图记录数据集。数据集中的文件定义为：

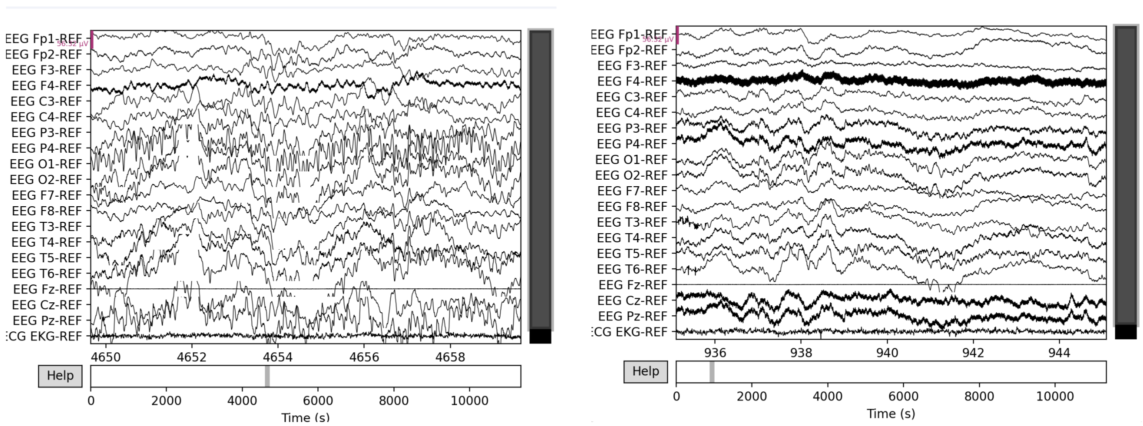
脑电图（1 至 79）：包含脑电图记录的 EDF 文件，以 256 Hz 采样。EEG 单位为微伏。

annotations_2017：CSV 文件或 MAT 文件，其中包含 3 位专家对每秒（1 Hz）采样的所有 79 名新生儿的注释。

clinical_information：CSV 文件包含从患者笔记中获取的临床数据以及与 EDF 文件对齐的记录中一致癫痫发作的一般空间分布信息（仅由审阅者 A 标记）。

读取EDF文件，得到如下EEG图。

Файл EDF считывался для получения следующей карты ЭЭГ.



(a) Части приступа

(b) Часть, не подверженная приступам

Рис. 1: график ээг сигналов

Моменты приступов в этих данных были получены из аннотации. Для анализа сигнала была выбрана часть приступа. Сигналы из каждого канала были нанесены на график.

选取癫痫发作的部分作分析。将各通道信号绘制到一张图上。

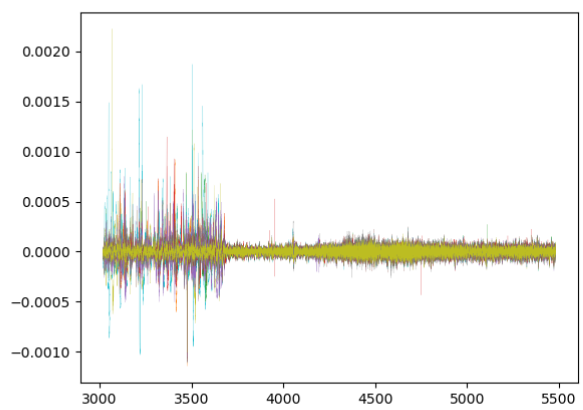


Рис. 2: Сигналы каждого канала в момент приступа

Усредняет все каналы в один сигнал.

将所有通道平均为一个信号。

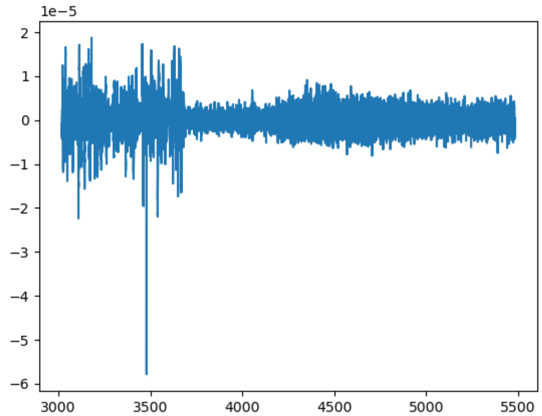


Рис. 3: усреднённый сигнал

Фильтрация сигналов с частотой выше 60 Гц.

过滤掉频率高于60Hz的信号。

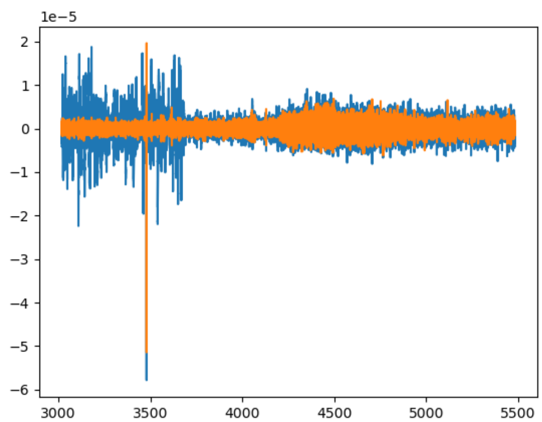


Рис. 4: Сигнал до и после фильтрации

2.2 Преобразование Фурье сигнала

Для отфильтрованного сигнала получаем спектрограмму, а также спектральную плотность мощности соответствующей полосы частот.

对于过滤后的信号，得到频谱图，以及对应频段的功率谱密度。

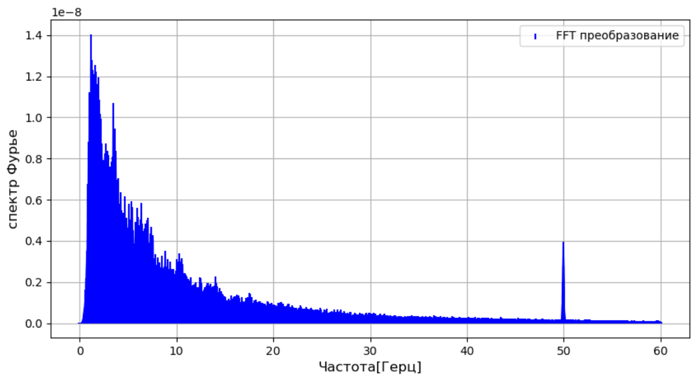


Рис. 5: спектрограмма

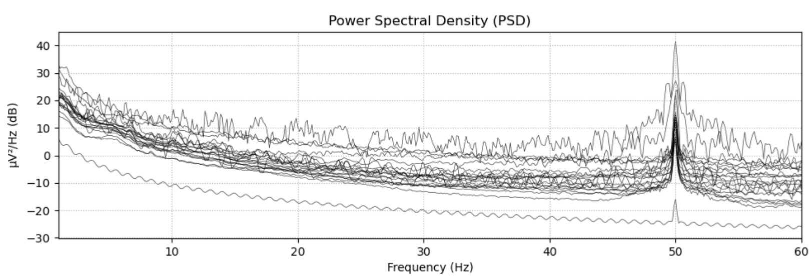


Рис. 6: спектральная плотность мощности

Рассчитать долю энергии в диапазонах Дельта (1-4 Гц), Тета (4-8 Гц), Альфа (8-12 Гц), Бета (12-30 Гц), Гамма (30-60 Гц).

计算Delta（1-4Hz）、Theta（4-8Hz）、Alpha（8-12Hz）、Beta（12-30Hz）、Gamma（30-60Hz）频段的能量占比。

2.3 Вейвлет-преобразование сигнала

В вейвлет-анализе мы используем вейвлет-функции для разложения сигналов и понимания их характеристик на разных масштабах и частотах.

在小波分析中，我们使用小波函数来分解信号，理解信号在不同尺度和频率上的特征。

Morlet小波是一种特定形式的小波函数，它通常用于在时间-频率域

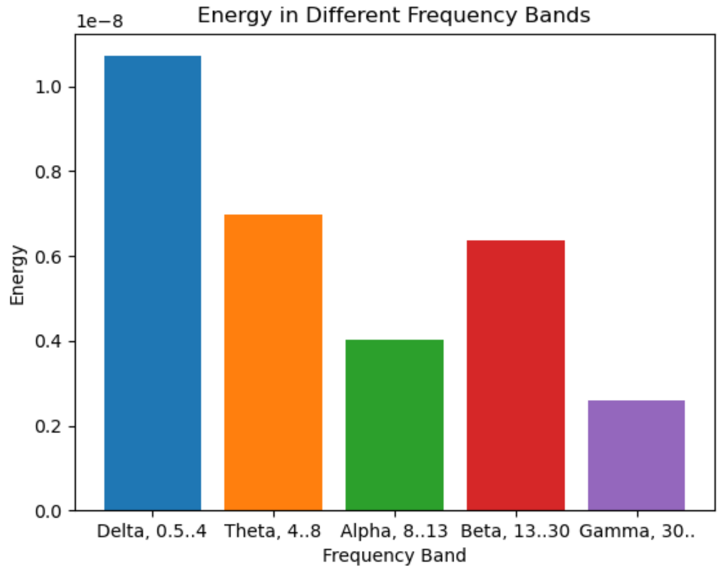


Рис. 7: Доля энергии в различных частотных диапазонах

Вейвлет Морлета - это особая форма вейвлет-функции, которая обычно используется для анализа сигналов во временно-частотной области. Выполняя вейвлет преобразование сигнала, мы можем получить ряд вейвлет-коэффициентов, которые отражают компоненты сигнала на разных масштабах и частотах.

Графики вейвлет-анализа в основном включают результаты непрерывного вейвлет преобразования, графики временных масштабов, частотно временные графики и графики вейвлет-дисперсии.

В вейвлет-анализе под «масштабом» обычно понимаются свойства вейвлет-базисных функций во времени и частоте. Базисная функция вейвлета используется для анализа основной формы сигнала, а мас-

中分析信号。通过对信号进行小波变换，我们可以获得一系列小波系数，这些系数反映了信号在不同尺度和频率上的成分。

小波分析图主要包括连续小波变换结果、时间-尺度图、频率-时间图和小波方差图。

在小波分析中，"尺度"通常是指小波基函数在时间和频率上的特性。小波基函数是用来分析信号的基本波形，而尺度描述了这个基本波形的展开或收缩程度。尺度越小，小波基函数在时间上的展开就越宽，频率上的局部化就越好；尺度越大，小波基函数在时间上的展开就越窄，频率上的局部化就越差。

在小波变换中，尺度和频率之间有一个反比关系。较小的尺度对应于较高的频率，而较大的尺度对应于较低的频率。这使得小波变换能够同时提供关于信号的时间和频率特征的信息，而不像其他变换（如

штаб описывает, насколько расширяется или сжимается эта основная форма сигнала. Чем меньше масштаб, тем шире разброс вейвлет-базисной функции во времени и тем лучше локализация по частоте; чем больше масштаб, тем меньше разброс вейвлет-базисной функции во времени и тем хуже локализация по частоте.

В вейвлет-преобразовании существует обратная зависимость между масштабом и частотой. Меньшие масштабы соответствуют более высоким частотам, а большие масштабы - более низким частотам. Это позволяет вейвлет преобразованию предоставлять информацию о временных и частотных характеристиках сигнала без компромисса между временным и частотным разрешением, который требуется при использовании других преобразований, таких как преобразование Фурье.

傅里叶变换）那样需要权衡时间和频率分辨率。

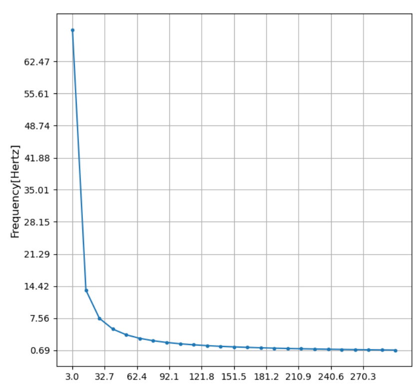


Рис. 8: Зависимость от частоты и масштаба

Вейвлет-скалограмма получается с помощью вейвлет-преобразования,

通过小波变换得到小波尺度图，以及小波方差图。

а вейвлет-дисперсия изображения.

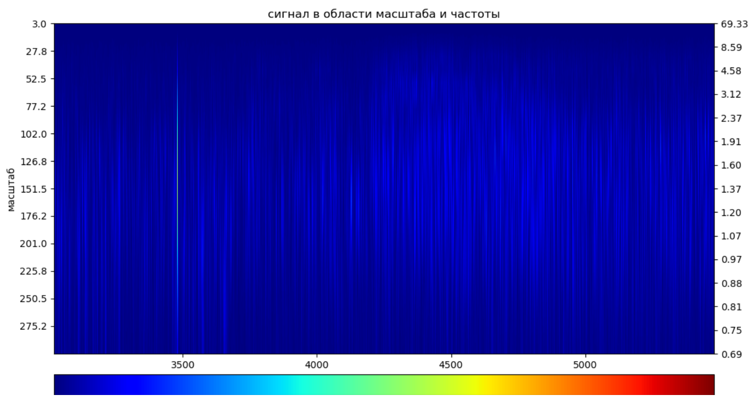


Рис. 9: Скейлограмма

Как показано на рисунке, во время приступа колебания амплитуды уровня меньше на шкале высоких частот и более выражены на шкале низких частот, особенно в момент приступа. На графике дисперсии наибольшая дисперсия наблюдается на шкале 150, что указывает на то, что уровень колеблется наиболее значительно на этой шкале, что соответствует графику шкалы выше.

如图所示，在癫痫发作期间，在高频尺度上，电平振幅波动较小，而在低频尺度上，电平振幅波动较为明显，特别是在癫痫发作的时刻。在方差图中，在尺度为150时的方差最大，表示电平在这个尺度上的波动最为明显，与上面的尺度图相吻合。

3 Ссылки на литературу

https://github.com/TAUforPython/wavelets/blob/main/wavelets_cwt_dwt_example_EEG_ECG.ipynb
https://blog.csdn.net/weixin_44259522/article/details/135138383
https://www.nature.com/articles/sdata201939

4 Код

```
import mne
import pywt
import numpy as np
```

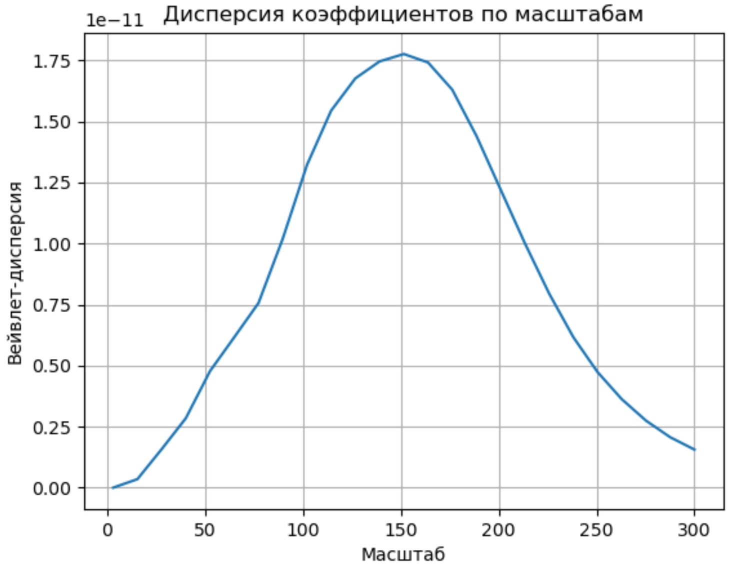


Рис. 10: Дисперсия-вейвлет

```
import matplotlib.pyplot as plt
import pandas as pd
from mne.preprocessing import ICA
from mne.datasets import eegbci

raw = mne.io.read_raw_edf('./lab4/eeg66.edf', preload=True)
raw.plot(scalings = 'auto', show = False)
plt.show()
annotations = raw.annotations
print(raw.ch_names)

eeg_data, eeg_times = raw.get_data(return_times=True)
print('number of reports in the time series:',len(eeg_times))
print(annotations)
t_index_begin = np.where(eeg_times > 1*3500 -480)[0][0]
t_index_end = np.where(eeg_times > 1*5000 + 480)[0][0]
t = eeg_times[t_index_begin:t_index_end]
T = t[-1] - t[0]
N = len(t)
eeg_data = np.delete(eeg_data, [len(eeg_data)-1], axis = 0)

for i in range(len(raw.ch_names)-2):
```

```
    plt.plot(t, eeg_data[i, t_index_begin:t_index_end],linewidth = 0.1)
plt.show()

y=[]
for i in range(len(eeg_data)-2):
    y =+ eeg_data[i, t_index_begin:t_index_end]
y =y/(len(eeg_data)-1)
plt.plot(t,y)
plt.show()

from scipy import signal
def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
    nyquist = 0.5 * fs
    low = lowcut / nyquist
    high = highcut / nyquist
    b, a = signal.butter(order, [low, high], btype='band')
    filtered_data = signal.lfilter(b, a, data)
    return filtered_data

fs = len(t)/(T)
lowcut = 1
highcut = 60
y_filt = np.apply_along_axis(butter_bandpass_filter, axis=0, arr=y, lowcut=lowcu
plt.plot(t,y, t, y_filt)
plt.show()
y = y_filt

from scipy.fft import fft, fftfreq
window = np.hanning(N)
y_windowed = y * window
yfft = fft(y_windowed)

xf = fftfreq(N, T/N)[:N//2]
yf = np.abs(yfft[0:N//2]) / N

f_viewmax = 60
if len(np.where(xf > f_viewmax)[0]) == 0:
  index_f_viewmax = len(xf)
else:
  index_f_viewmax = np.where(xf > f_viewmax)[0][0]
print('number of points in the range of Fourier transform:',len(xf))
print('maximum frequency', xf[-1])
plt.figure(figsize=(10, 5))
plt.stem(xf[0:index_f_viewmax], yf[0:index_f_viewmax],
        linefmt='b-', markerfmt=' ', basefmt=' ', label='FFT transformation')
plt.xlabel('Frequency[Hz]', fontsize=12)
```

```
plt.ylabel('Fourier spectrum', fontsize=12)
plt.grid()
plt.legend()
plt.show()

raw.plot_psd(fmin=1, fmax=f_viewmax, tmax=np.inf, show=False)
plt.title('Power Spectral Density (PSD)')
plt.show()

bands = {'Delta': (1, 4), 'Theta': (4, 8), 'Alpha': (8, 12),
         'Beta': (12, 30), 'Gamma': (30, 60)}
for band, (f_low, f_high) in bands.items():
    mask = (xf >= f_low) & (xf <= f_high)
    power = np.trapz(yf[mask], xf[mask])
    print(f"{band} Band Power: {power:.2f}")
    plt.bar(band, power)

plt.title('Energy in Different Frequency Bands')
plt.xlabel('Frequency Band')
plt.xticks(np.arange(5), ('Delta, 0.5..4', 'Theta, 4..8', 'Alpha, 8..13', 'Beta,
plt.ylabel('Energy')
plt.show()

scale_max = 300
scale_min = 3
scales = np.linspace(scale_min, scale_max, num=25, endpoint=True)
# scales = np.logspace(np.log10(scale_min), np.log10(scale_max), num = 25, endpo

wavelet_core = 'morl'
# dt = t[1] - t[0]
fs = len(t) / (T)
dt = 1 / fs
coef, freqs = pywt.cwt(y, scales, wavelet_core, sampling_period=dt)

f = pywt.scale2frequency(wavelet_core, scales)/dt

plt.figure(figsize=(7, 7))
plt.grid()
plt.yticks(np.arange(min(freqs), max(freqs), (max(freqs) - min(freqs))/10))
plt.xticks(np.arange(min(scales), max(scales), (max(scales) - min(scales))/10))
plt.ylabel('Frequency[Hertz]', fontsize=12)
plt.xlabel('scale in wavelet transform', fontsize=12)
plt.plot(scales, freqs, '.-')
plt.show()

plt.figure(figsize=(10, 7))
```

```
ax1 = plt.subplot(211)
#plt.imshow(abs(coef), extent=[t[0], t[-1], scale_max, 0], interpolation='biline
plt.imshow(abs(coef), cmap='jet', aspect='auto', extent=[t[0], t[-1], max(scales
plt.gca().invert_yaxis ()
plt.ylabel('scale ', fontsize=12)

ax2 = plt.subplot(212, sharex=ax1)
plt.plot(t, y)
ax2.set_title("time domain signal")

#Plotting scalogram
plt.figure(figsize=(10, 5))
plt.imshow(abs(coef), extent=[t[0], t[-1], max(scales), min(scales)], interpolat
plt.gca().invert_yaxis ()
plt.yticks(np.arange(min(scales), max(scales), (max(scales) - min(scales))/10))
plt.ylabel('scale ', fontsize=12)
plt.xlabel('time ', fontsize=12)
plt.show()

from matplotlib.figure import cbar
from matplotlib.ticker import FuncFormatter
import mpl_toolkits.axes_grid1 as axes_grid1
import matplotlib.ticker as ticker

#Plotting dual axis scalogram
f1 = plt.figure()
f1.set_size_inches(12, 6)


ax1 = axes_grid1.host_axes([0.1, 0.1, 0.8, 0.80])
axc = f1.add_axes([0.1, 0, 0.8, 0.05])
im1 = ax1.imshow(abs(coef), cmap='jet', aspect='auto', interpolation='bilinear',
                 extent=[t[0], t[-1], max(scales), min(scales)],
                 vmax=abs(coef).max(), vmin=-abs(coef).min())

cbar.Colorbar(axc, im1, orientation='horizontal')

#ax1.invert_yaxis()
ax1.set_yticks(np.arange(min(scales), max(scales), (max(scales) - min(scales))/1


ax2 = ax1.twinx()
# make ticklabels on the top invisible
ax2.axis["top"].toggle(ticklabels=False)
```

```
formatter = FuncFormatter(lambda x, pos: '{:0.2f}'.format(pywt.scale2frequency(w
ax2.yaxis.set_major_formatter(formatter)
ax2.set_ylim(ax1.get_ylim())
#ax2.invert_yaxis()

# make number ticks what we want
#ax2.yaxis.set_major_locator(ticker.MultipleLocator(0.5))
ax2.yaxis.set_major_locator(ticker.LinearLocator(numticks = 15))


ax2.set_ylabel('frequency')
ax1.set_ylabel('scale')
ax2.set_xlabel('time')

ax1.set_title("signal in the scale and frequency domain")

plt.show()
dispersion = np.var(np.abs(coef), axis=1)

plt.plot(scales, dispersion)
plt.xlabel('Scale')
plt.ylabel('Wavelet variance')
plt.title('Variance of coefficients by scale')
plt.grid()
plt.show()
```