Python 连续小波分析

Dezeming Family

2022 年 5 月 13 日

DezemingFamily 系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列书，可以从我们的网站 [https://dezeming.top/] 找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

目录

一 PyWavelets 简介 **1**

二 函数介绍 **1**

2 1 连续小波变换函数 1

2 2 尺度如何转化为频率 3

三 代码测试 **3**

3 1 数据生成 3

3 2 根据中心频率定义尺度 3

参考文献 **4**

一 PyWavelets 简介

PyWavelets 是 Python 小波变换程序，而且是开源的程序，Github 的源代码可以参考 [1]，官方文件可以参考 [5]。该代码的主要维护人是：

- Gregory R. Lee [2]
- Ralf Gommers [3]
- Filip Wasilewski [4]
- Kai Wohlfahrt
- Aaron O’Leary

PyWavelets 支持下面多种功能：

- 一维到多维的离散小波变换和逆变换。
- 一维到多维的多尺度小波变换与逆变换。
- 一维到多维的平稳小波变换（非抽取小波变换）。
- 一维和二维小波包分解与重构。
- 一维连续小波变换。
- 小波和尺度函数的近似计算。
- 100 多个内置小波滤波器，支持自定义小波。
- 支持单精度和双精度计算。
- 支持实数和复数的计算。
- 结果与 Matlab 小波工具箱（TM）兼容。

不同的功能索引可以参考 [6]。

以前本科期间，我主要使用的是 matlab 小波工具箱，使用小波分析也只是为了一些创新创业项目，当时并没有小波分析课程，自己也没有较为深入地学习小波分析。当前，我在 Pywt 上用到的功能没有那么多，现在主要以快速上手为准，其他的一些方面以后我会再补充。

二 函数介绍

虽说是连续小波变换，但其实处理的也是离散信号，只是区别于离散小波变换，连续小波变换是直接对信号做小波变换，在不同的尺度和空间求值，而不是做塔式分解。

2 1 连续小波变换函数

调用以下代码，我们可以查看有哪些可以用来做连续小波分析的小波：

```
wavlist = pywt.wavelist(kind='continuous')
print(wavlist)
```

输出为：

```
['cgau1', 'cgau2', 'cgau3', 'cgau4', 'cgau5', 'cgau6', 'cgau7', 'cgau8', '
    cmor', 'fbsp', 'gaus1', 'gaus2', 'gaus3', 'gaus4', 'gaus5', 'gaus6', '
    gaus7', 'gaus8', 'mexh', 'morl', 'shan']
```

看到有人问如果使用 db 系列小波做连续小波变换为什么会出错，这是因为其实 db 小波并没有被定义为连续小波。做连续小波变换的小波不一定是正交小波，只要满足小波的条件，就可以用来做连续小波分析。

做连续小波分析需要用到 pywt.cwt 函数的参数列表为：

```
coefs, frequencies = cwt(data, scales, wavelet, sampling_period=1., method='conv', axis=-1)
```

data 是 array 类型的输入信号，wavelet 是使用的小波名称，例如'wavelet='cgau8"。

method 是计算方式，有两种，一种是使用 FFT 做频域卷积，另一种是使用 conv 做时域卷积。对于比较大的信号来说，频域卷积更快一些，而对于比较小的信号，时域卷积更快。因此，我们还可以选择 auto，表示程序自动计算，帮我们选择频域卷积还是时域卷积。

scales 和 sampling_period 相互关联：

```
f = scale2frequency(wavelet, scale)/sampling_period
```

scales 决定了小波被压缩或者被拉伸的程度。较低的 scales 值会压缩小波，更好地与高频相关。这里的 scales 是一个数组，表示一共分析多少个尺度。

sampling_period 表示对于频率输出的采样周期（可选），scales 并不会根据采样周期 sampling_period 来进行放缩。根据源码中的描述，sampling_period 仅仅在 cwt 程序的最后用到了一次，就是将要返回的 frequencies 全部除以 sampling_period（换句话说，好像没有任何用处）：

```
    frequencies = scale2frequency(wavelet, scales, precision)
    if np.isscalar(frequencies):
        frequencies = np.array([frequencies])
    frequencies /= sampling_period
```

scale2frequency 函数调用了 central_frequency 函数：

```
def scale2frequency(wavelet, scale, precision=8):
    return central_frequency(wavelet, precision=precision) / scale
```

central_frequency 函数求小波的中心频率，关于中心频率可以去 [7] 进行了解，我们在下一节也会详细解释一下。

返回的 coefs 数组的大小取决于输入 data 的长度和所给的 scales 的长度，这是一个二维数组，第一个维度与尺度有关，第二个维度与 data 大小有关。

```
# data的大小是[600,]
wavename = 'cgau8'
scales = np.arange(1,100)
coef, freqs = pywt.cwt(data, scales, wavename, 1)
print(coef.shape)
print(freqs.shape)
```

打印输出为：

```
(99, 600)
(99,)
```

axis 表示要变换的轴。因为我们传入的 data 可能是多维数据，因此需要选择一个轴来进行变换，比如传入的是二维数，这里的连续小波变换只能变换一维，因此需要选择横轴还是纵轴的数据来变换。如果不用默认的最后一个轴（axis=-1），则 cwt 函数通过调用 data.swapaxes 函数来将最后一个轴与 axis 轴交换，变换完以后再交换回来。

2 2 尺度如何转化为频率

我们知道小波并不像是傅里叶变换那样，不同的傅里叶基对应不同的频率（而且诶这个频率是固定的值）。一个小波基相当于一个带通滤波器，因此它的频率处于一个段上。但是我们知道，尺度越小的小波，它震荡的就越快，因此能感受到它的频率越高，为了在示意图上表现得更清楚，我们一般用中心频率来代指这个小波的频率。

中心频率就是最高频率加最低频率再除以 2 吗？显然不是。我们希望一个小波基在某段的频率占比大，则中心频率就越倾向于该频率，因此相当于加权求和，权重与该频率处的值大小有关。关于小波母函数的中心频率的计算可以参考其他的资料，这里不再赘述。

不同尺度下的小波中心频率为：

$$central\_frequency = \frac{frequencies\_motherWavelet}{scale \times sampling\_period} \tag{二.1}$$

也就是小波母函数的中心频率除以尺度值再除以采样周期（即乘以采样频率）。

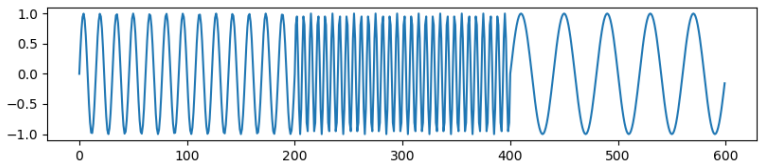
三 代码测试

3 1 数据生成

为了更清楚地看到效果，我们生成三种频率的正弦波：

```
aa = []
for i in range(200):
    aa.append(np.sin(0.3*np.pi*i))
for i in range(200):
    aa.append(np.sin(0.13*np.pi*i))
for i in range(200):
    aa.append(np.sin(0.05*np.pi*i))
plt.plot(aa)
plt.show()
```

显示图像为：



3 2 根据中心频率定义尺度

比如我们总共需要 256 个尺度：

```
wavename = 'cgau8'
sampling_rate = 1024
totalscal = 256
fc = pywt.central_frequency(wavename)
cparam = 2 * fc * totalscal
scales = cparam / np.arange(totalscal, 0, -1)
[cwtmatr, frequencies] = pywt.cwt(aa, scales, wavename, 1.0 / sampling_rate)
```

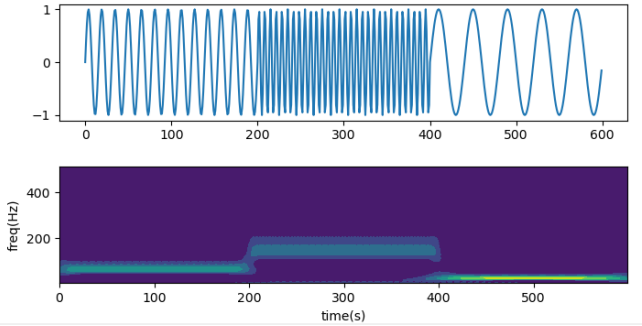
注意 np.arange 生成从 256 到 1 的 256 个整数。

我们希望这 256 个尺度跨越的频率区间是 [0, 2*fc*totalscal]，这个区间长度为 cparam。

显示一下结果：

```
plt.figure(figsize=(8, 4))
plt.subplot(211)
t = np.arange(0, dataSize, 1.0)
plt.plot(t, aa)
plt.subplot(212)
plt.contourf(t, freqs, abs(coef))
plt.ylabel(u"freq(Hz)")
plt.xlabel(u"time(s)")
plt.subplots_adjust(hspace=0.4)
plt.show()
```

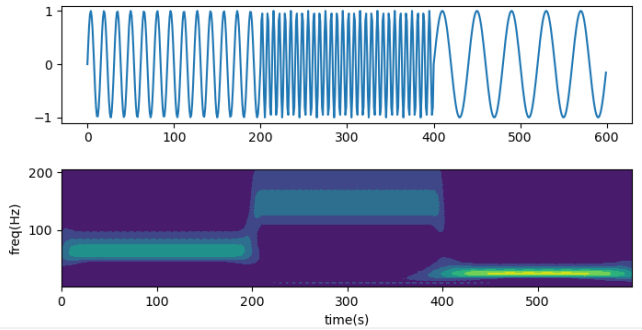
得到输出显示：



当设置：

```
cparam = 5 * fc * totalscal
```

输出显示：



所以也可以这么理解，当不同的 scale 之间差距变小时，相当于计算的分辨率提高，显示更精细。

参考文献

[1] https://github.com/PyWavelets/pywt

[2] https://github.com/grlee77

[3] https://github.com/rgommers

[4] http://en.ig.ma/

[5] https://pywavelets.readthedocs.io/en/latest/

[6] https://pywavelets.readthedocs.io/en/latest/ref/index.html

[7] https://ww2.mathworks.cn/help/wavelet/ref/centfrq.html