信号処理システム特論
最終レポート

21312885
NGUYEN ANH QUAN

2023 年 07 月 22 日

1　作成したプログラム

View source code on github

```
from scipy.io import wavfile
import numpy as np
import scipy.signal as sp
import scipy as scipy
import matplotlib.pyplot as plt
import pandas as pd
import soundfile as sf
import museval
```

```
x1, fs1 = sf.read('x1.wav')
x2, fs2 = sf.read('x2.wav')
x3, fs3 = sf.read('x3.wav')
s1, fs_s1 = sf.read('s1.wav')
s2, fs_s2 = sf.read('s2.wav')
s3, fs_s3 = sf.read('s3.wav')
```

```
# 標準化
M = len(x1)信号のサンプル数#
x = np.zeros((3, len(x1)))
x[0,:] = (x1 - np.mean(x1))/np.sqrt(np.nanvar(x1))
x[1,:] = (x2 - np.mean(x2))/np.sqrt(np.nanvar(x2))
x[2,:] = (x3 - np.mean(x3))/np.sqrt(np.nanvar(x3))
print(x.shape)

# 観測信号の白色化化
R = np.dot(x, x.T)/M
eig_v, Q = np.linalg.eig(R)
Lambda = np.diag(eig_v)
V = np.dot(np.sqrt(np.linalg.inv(Lambda)), Q.T)
xh = np.dot(V,x)

b_past = np.array(([1,1,1],[0,0,0],[0,0,0]),dtype=float)
b = np.array(([1,1,1],[0,0,0],[0,0,0]),dtype=float)
print(b.shape)

def ICA(m_component, max_iter):
    for m in range(m_component):
        for i in range(max_iter):
```

```
            b_past[:,m] = b[:,m]
            sh = np.dot(b[:,m].T, xh)
            # fourth moment - kurtosis
            b[:,m] = np.mean(xh * np.power(sh,3).reshape(1, -1), axis=1) - 3*b[:,m]

            # Gram-Schmidt process
            if m > 0:
                for j in range(m):
                    b[:,m] -= np.dot(np.dot(b[:,m].T, b[:,j]), b[:,j])

            # Normalize
            b[:,m] /= np.linalg.norm(b[:,m])

            if np.abs(np.abs(np.dot(b[:,m].T, b_past[:,m])) - 1) < 10**-6:
                break
        # print("iteration = ",i)
    return np.dot(b.T, xh)

sh = ICA(3,100)
s = np.vstack((s1,s2,s3))
print(sh.shape,xh.shape)
```

```
def z_score(x):
    new_val = []
    for i in range(3):
        new_val = np.append(new_val, (x[i,:] - np.mean(x[i,:]))/np.std(x[i,:]))
    new_val = new_val.reshape(sh.shape[0],sh.shape[1])
    return new_val
s_normalized = z_score(s)
#Sh already have mean = 0, and std = 1
```

```
t = np.linspace(1,10,fs_s1*10)
s_title = ["s1, sh1","s2, sh2","s3, sh3"]
# sh_title = ["sh1","sh2","sh3"]
fig, axs = plt.subplots(1,3,figsize=(19, 7),sharex= True)
for i in range(3):
        axs[i].plot(t, s_normalized[i,:], label='S{}'.format(i+1))
        axs[i].plot(t, sh[i,:], label='Sh{}'.format(i+1))
        axs[i].set_title(s_title[i])
for ax in axs.flat:
    ax.set(xlabel='time[s]', ylabel='Amplitude')
    ax.legend()
fig.suptitle('The Source signals and The recovered signals after z-score Normalized ')
```

```
signal_power = np.var(s_normalized, axis=1)   # Power of original signal
noise_power = np.var((s_normalized - sh),axis=1)
print(noise_power)
snr = 10*np.log10(signal_power/noise_power)
print("SNR : ",snr)
```

```
MSE = []
for i in range(3):
    MSE = np.append(MSE,np.sum(np.power((s_normalized[i,:] - sh[i,:]),2))/(sh.shape[1]))
print('MSE :',MSE)
```

```
SDR,ISR,SIR,SAR = museval.evaluate(s_normalized,sh)
print('SDR :',np.mean(SDR,axis=1))
print('ISR :',np.mean(ISR,axis=1))
```

```
print('SIR :',np.mean(SIR,axis=1))
print('SAR :',np.mean(SAR,axis=1))
```

```
s_sh = np.concatenate((s,sh),axis= 0).T
print(s_sh.shape)
df = pd.DataFrame(s_sh, columns = ['s1','s2','s3','sh1','sh2','sh3'])
corr_mat = df.corr(method='pearson')
import seaborn as sns
sns.heatmap(corr_mat,
            vmin=-1.0,
            vmax=1.0,
            center=0,
            annot=True,
            fmt='.2f',
            xticklabels=corr_mat.columns.values,
            yticklabels=corr_mat.columns.values
           )
plt.show()
```

```
t = np.linspace(1,10,fs_s1*10)
s_title = ["s1","s2","s3"]
sh_title = ["sh1","sh2","sh3"]
fig, axs = plt.subplots(2, 3,figsize=(20, 12),sharex= True)
for i in range(2):
    if (i == 0):
        y = s
        title = s_title
    else:
        y = sh
        title = sh_title
    for j in range(3):
        axs[i, j].plot(t, y[j,:])
        axs[i, j].set_title(title[j])
for ax in axs.flat:
    ax.set(xlabel='time[s]', ylabel='Amplitude')
```

2 分離信号と元信号の相関行列

分離信号と元信号の相関行列は以下に示す。

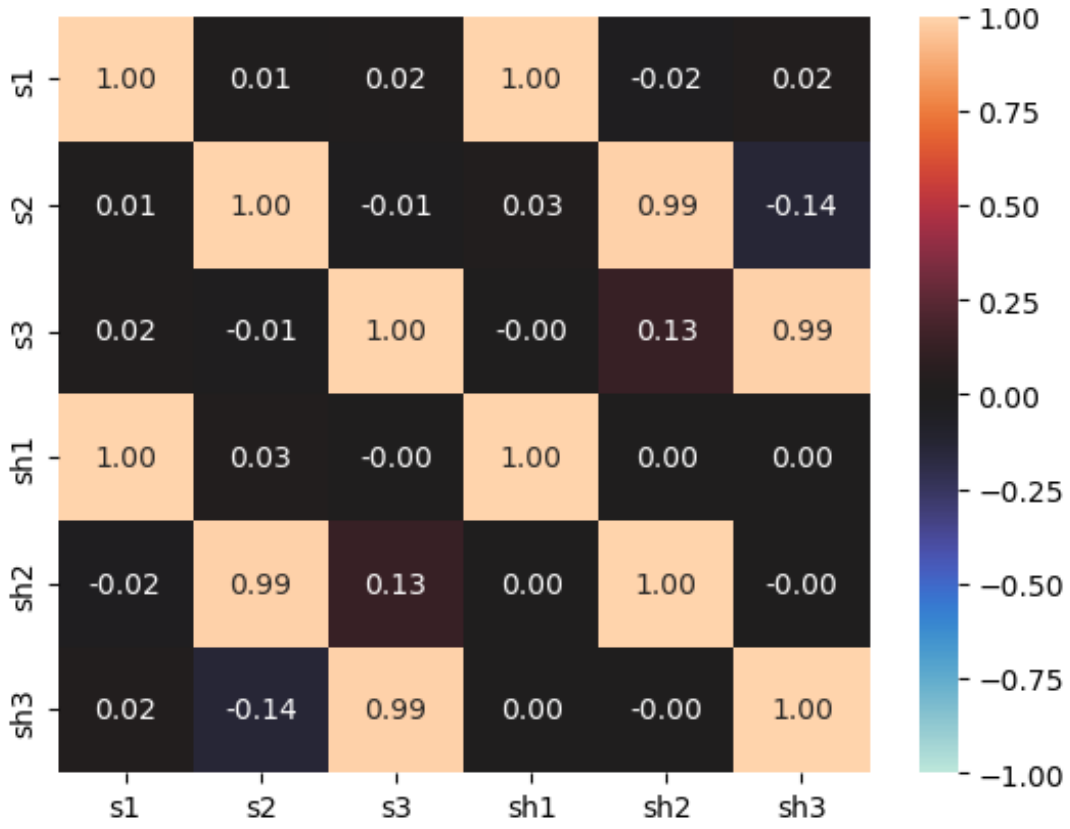

Fig.1: Correlation matrix of separated signal and the original signal

ここでは、ピアソン相関を使用して、分離信号と元信号の相関の程度を評価した。
ピアソンの相関係数の範囲は -1 から 1 である。相関係数が正の場合、変数間は正の線形関係を持ち、相関係数が負の場合、変数間は負の線形関係を持ち、相関係数が 0 の場合、変数間は互いに関係がない（相関なし)。また、相関係数は ± 1 に近づくほど、相関の程度が強くなる。
相関行列から、s1 と sh1 の間、s2 と sh2 の間、s3 と sh3 の間の相関はすべて正の線形関係であることがわかった。 ここで、s1 と sh1 の間の相関は完全な正の線形関係である。 つまり、s1 が増加すると、sh1 も比例に増加し、s1 が減少すると、sh1 も比例に減少した。

3　分離信号と対応する元信号の図

分離信号と対応する元信号の図は以下に示す。

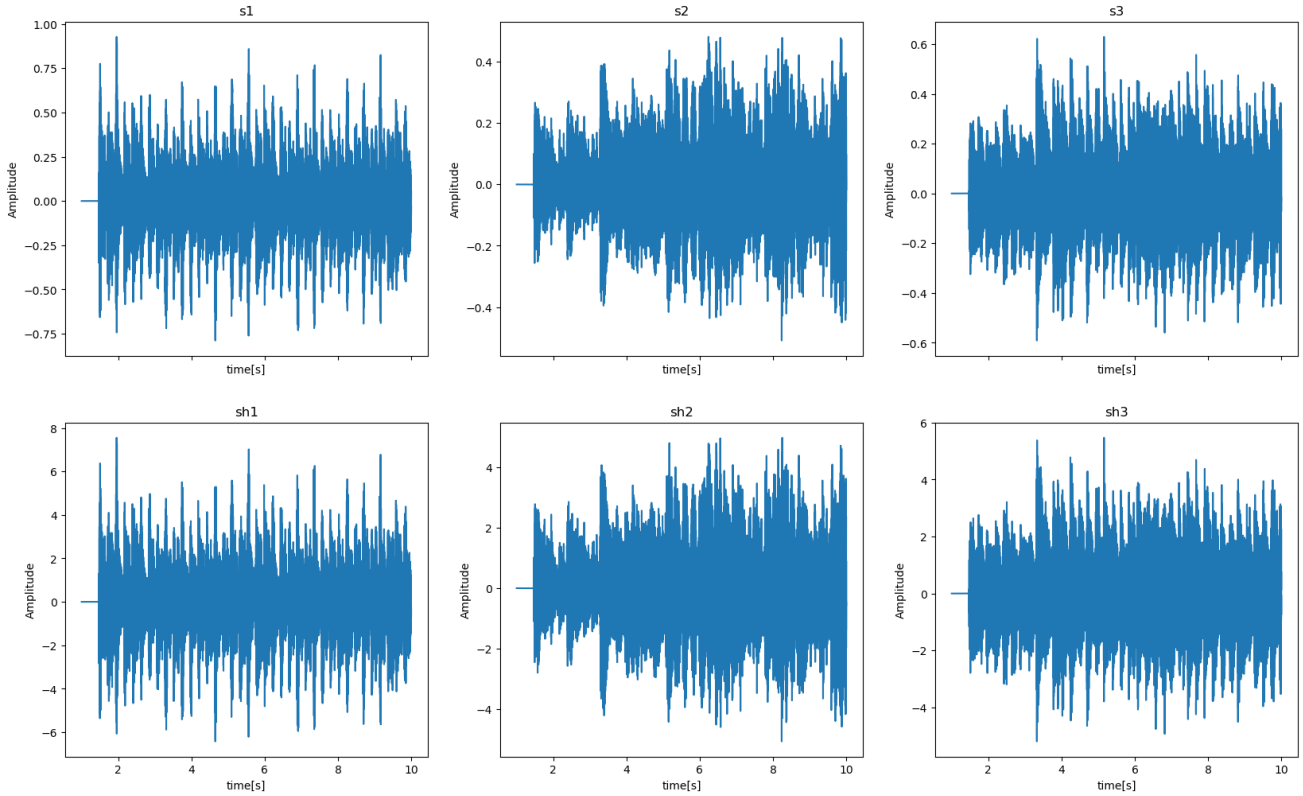

Fig.2: Separated Signals and Corresponding Original Signals

4 分離信号と元信号との定量的評価値

- SIR（Source to Interference Ratio): SIR は、分離信号に存在する干渉（interference）の量を定量化するために使用された。 ここで、干渉（interference）とは、他のソースからの信号が ICA を介して分離信号に存在することを意味する。
  SIR 値が高いほど、分離信号での干渉量が少なく、分離信号の質が高くなっている (元信号に似ている)。
- SIR の式 [1]：

$$SIR_j = 10\log_{10}\left(\frac{\sum_{i=1}^{I}\sum_t (s_{ij}^{img}(t) + e_{ij}^{spat}(t))^2}{\sum_{i=1}^{I}\sum_t e_{ij}^{interf}(t)^2}\right) \tag{1}$$

ここで、

j　元信号の数（このレポートで 3 つ）
i　信号のチャンネル数（このレポートで 1 つ）
$s_{ij}^{img}(t) + e_{ij}^{spat}(t)$　ターゲットソースからの信号
$e_{ij}^{interf}(t)$　分離信号に存在する干渉（interference）
また、このレポートではチャネル数が 1 であり、空間内の音の伝播やスピーカーの配置については言及されていないため、空間歪み $e_{ij}^{spat}(t)$ は 0 と仮定される。そこで、SIR の式は次のように書き換える。

$$SIR_j = 10\log_{10}\left(\frac{\sum_t s_{ij}^{img}(t)^2}{\sum_t e_{ij}^{interf}(t)^2}\right) \tag{2}$$

式 (2) は、元信号のパワーに対する分離信号に存在する干渉パワーの比として理解できた。単位は dB

である。

- SIR 指標を採用した理由：
  分離信号は ICA を使用して観測信号から分離した。観測信号は、3 つの元信号を混合して形成された。そのため、いずれか 1 つのソースの分離信号には、他の 2 つのソースからの信号成分が含まれる可能性がある。したがって、分離信号と元信号との定量的評価するには、SIR 指標を採用して分離信号での干渉量を定量することが最も効果的だと思います。
  3 つのソースの SIR[dB] 指標の結果は、それぞれ 30.81, 16.67, 17.91 である。結果によると、3 つの分離信号 Sh1、Sh2、Sh3 のうち、分離信号 Sh1 の質が最も良く、最も信号質が低いのは Sh2 である。

参考文献

[1] E. Vincent, H. Sawada, P. Bofill, S. Makino, and J. P. Rosca, *First stereo audio source separation evaluation campaign: data, algorithms and results*, ICA 2007, pp. 552-559, Sep. 2007.