プログラミング基礎演習レポート 2018 —

尖度最大化に基づいての独立成分分析 (ICA)をPythonで実装

学籍番号：03189108

名前：王昊中

I. 導入

独立成分分析（ICA）は、多変量の信号を複数の加法的な成分に分離するための計算手法である。各成分は、ガウス的でない信号で相互に統計的独立なものを想定する。ICAモデルは一般性が高いので、多くの分野で応用されている。例えば、

①脳機能の可視化では、多くの場合脳中にいくつかの信号源があり、それかあら出る信号画の雑多者が頭外のセンサに現れる。

②計量経済学では並列的な時系列がよく出てくる。ICARDSによってそれらを道立な成分に分解すると、データの構造に対する洞察を得られるこｔがある

③多少異なる応用として画像の特徴抽出がある。

ICAが扱える問題はこのように設定する。

観測データを$\boldsymbol{x} \in \mathrm{R}n$で表し，信号源データ(未知)を$\boldsymbol{y} \in \mathrm{R}n$とする(どちらも列ベクトル).実際 のデータは$\boldsymbol{X} = \{\boldsymbol{x}1,\boldsymbol{x}2,...\boldsymbol{x}N\}$のように$N$個のデータベクトルによって与えられ，それに対応して信号 源は$\boldsymbol{Y} = \{\boldsymbol{y}1, \boldsymbol{y}2, \ldots , \boldsymbol{y}N\}$とする.ここでは，観測データ$\boldsymbol{x}$は平均が$\boldsymbol{0} \in \mathrm{R}n$になるように調整している(つまり$\boldsymbol{x} \leftarrow \boldsymbol{x} - \mathbb{E}[\boldsymbol{x}]$).なお，$\mathbb{E}[\boldsymbol{x}]$は平均(期待値)を表す。観測データは，信号源の重ね合わせのため，以下の線形関係があると仮定する

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{y}(1)$$

ここで$\boldsymbol{A} \in \mathrm{R}^{n \times n}$の行列である.これより，もし$\boldsymbol{W} = \boldsymbol{A}^{-1}$が分かれば，信号源$\boldsymbol{y}$は

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} \quad (2)$$

によって復元可能である.しかし，たった二つの確率変数q_1とq_2を足した場合でも，足した後の$q_1 + q_2$の方が，q_1やq_2よりも正規分布に近い分布となる場合が多い.実は，尖度 (kurtosis)と呼ばれる統計量は，正規分布の時 0 になり，正規分布から乖離すると 0 から大きくずれる(尖度は正負の値をとれる).平均が0の確率変数$y \in \mathrm{R}$の尖度は以下で定義される.

$$\mathrm{kurtosis}[y] = \mathbb{E}[y^4] - 3\mathbb{E}[y^2]^2 \quad (3)$$

つまり，|kurtosis(y)|を最大化するように，$\boldsymbol{W}$を求めれば ICA が実現できることがわかる.

II. 手法・結果

（課題１を例にする、他の課題はほぼ同じである）

１．まず、ライブラリを導入する

```
import random
from numpy import *
import matplotlib.pyplot as plt
import copy
```

２．次に、データを導入する（画像のflatten化も必要である）

```
def load_data():
    x1=loadtxt('dat1.txt')
    x2=loadtxt('dat2.txt')
    x1=x1-mean(x1)
    x2=x2-mean(x2)
    X=array([x1,x2])
    return X;
```

３．白色化と呼ばれる操作を，ICA の前段階に挟む必要がある．

```
def whiten(X):
    xx=dot(X,X.T)
    D,E = linalg.eig(xx)
    D=diag(1/sqrt(D))
    V = dot(D, E.T)
    Z=dot(V,X)
    return Z
```

４．最後に以下の繰り返しアルゴリズムを適用することで可能である（少し改良する）

1. W(0)に対して初期値を適当に選ぶ.k=1.正規化する$w(0) \leftarrow w(0)/\|\boldsymbol{w}(0)\|$

2. $w(k) \leftarrow \mathbb{E}[\boldsymbol{z}(w(k-1)^{\top}\boldsymbol{z})^{3}] - 3w(k-1)$を計算する

3. w(k) ←w(k)／‖$\boldsymbol{w}$(k)‖ によって正規化する

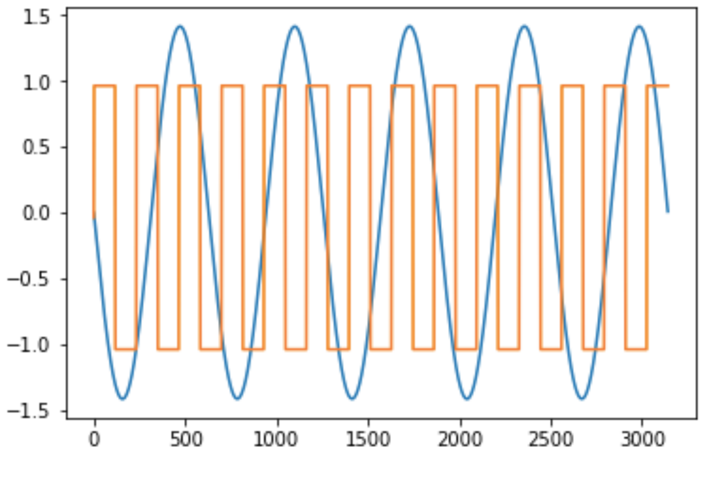
4. 相関除去

5. 収束していなければ，2 に戻る.(|w(k)$^{\top}$w(k-1)|==1)収束していれば終了する.

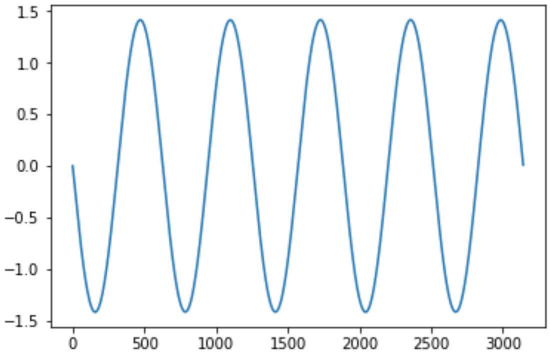
```
def Decorrelation(W):
    d, e = linalg.eigh(dot(W, W.T))
    return dot(dot(e * (1. / sqrt(d)), e.T), W)

def do_fastica(Z):
    n, m = Z.shape;
    Z *= sqrt(Z.shape[1])
    W = ones((n,n), float32)
    y=ones((n,m), float32)
    for i in range(n):
        for j in range(n):
            W[i][j] = random.random()
    for i in range(n):
        W[i]= W[i]/linalg.norm(W[i])
        while True:
            W1=copy.copy(W)
            y[i] = dot(W[i].T,Z)
            for j in range(n):
                W[i][j] = mean((Z*(y[i]**3))[j]) - 3*W[i][j]
            W[i] = W[i]/linalg.norm(W[i])
            W = Decorrelation(W)
            lim = abs(abs(dot(W[i].T,W1[i]))-1)
            if lim < 0.000000000001:
                break
    return W
```

5．結果

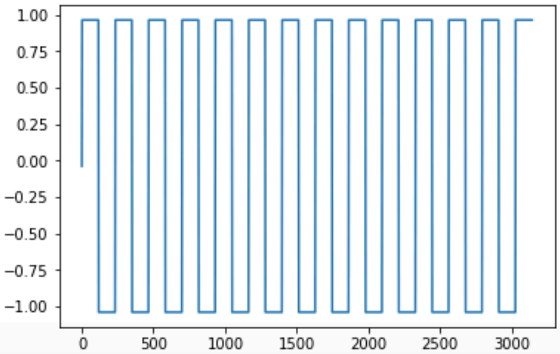
```
X=load_data()
Z=whiten(X)
W = do_fastica(Z)
S=dot(W,Z)
plt.plot(S[0])
plt.plot(S[1])
plt.show()
```



```
plt.plot(S[0])
plt.show()
```




```
plt.plot(S[1])
plt.show()
```



III．考察

1．尖度最大化に基づいての独立成分分析

もとのデータでは

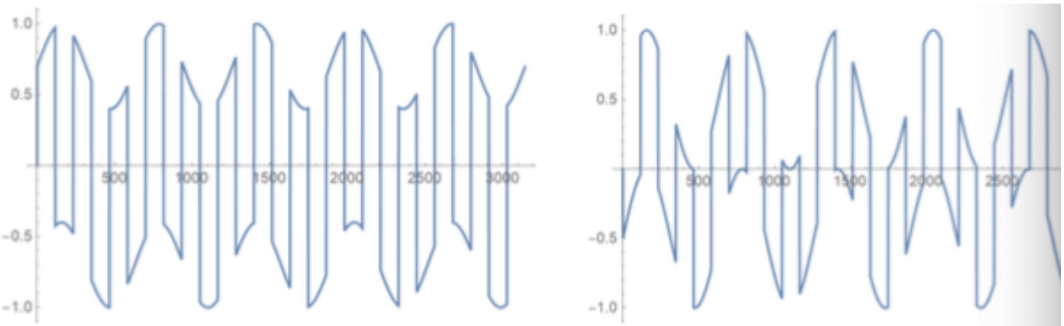
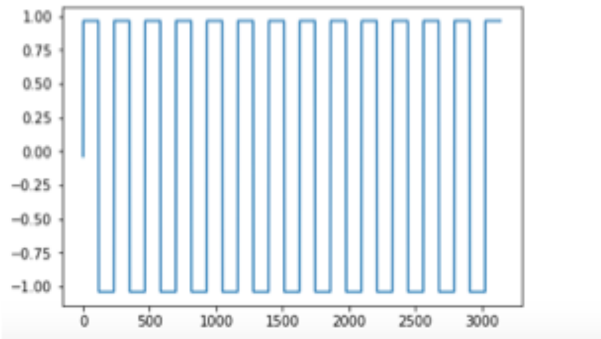
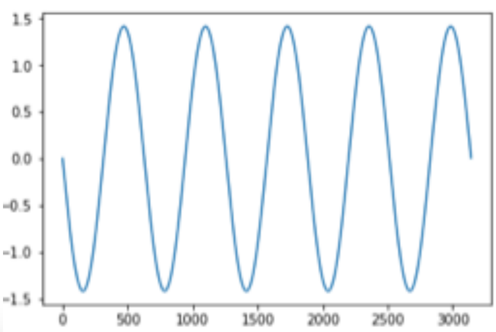


図1 ：テスト用のデータ．左から dat1.txt, dat2.txt.

処理したデータでは





きれいであるが、実は正確かどうかわからない。他の方法で確かめてみる

1. Center the data to make its mean zero.
2. Whiten the data to give $\mathbf{z}$.
3. Choose m, the number of independent components to estimate.
4. Choose initial values for the $\mathbf{w}_i$, $i = 1, ..., m$, each of unit norm. Orthogonalize the matrix $\mathbf{W}$ as in step 6 below.
5. For every $i = 1, ..., m$, let $\mathbf{w}_i \leftarrow E\{\mathbf{z}g(\mathbf{w}_i^T\mathbf{z})\} - E\{g'(\mathbf{w}_i^T\mathbf{z})\}\mathbf{w}$, where g is defined, e.g., as in (8.31)–(8.33).
6. Do a symmetric orthogonalization of the matrix $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_m)^T$ by

$$\mathbf{W} \leftarrow (\mathbf{W}\mathbf{W}^T)^{-1/2}\mathbf{W}, \tag{8.51}$$

   or by the iterative algorithm in Sec. 8.4.3.
7. If not converged, go back to step 5.

Table 8.4 The FastICA algorithm for estimating several ICs, with *symmetric* orthogonalization. The expectations are estimated in practice as sample averages.

```
def _logcosh(x, fun_args=None, alpha = 1):
    gx = tanh(alpha * x, x); g_x = gx ** 2; g_x -= 1.; g_x *= -alpha
    return gx, g_x.mean(axis=-1)

def do_decorrelation(W):
    s, u = linalg.eigh(dot(W, W.T))
    return dot(dot(u * (1. / sqrt(s)), u.T), W)

def do_fastica(Z):
    n, m = Z.shape; p = float(m); g = _logcosh
    Z *= sqrt(Z.shape[1])
    W = ones((n,n), float32)
    for i in range(n):
        for j in range(n):
            W[i,j] = random.random()
    maxIter = 200
    for ii in range(maxIter):
        gwtx, g_wtx = g(dot(W, Z))
        W1 = do_decorrelation(dot(gwtx, Z.T) / p - g_wtx[:, newaxis] * W)
        lim = max( abs(abs(diag(dot(W1, W.T))) - 1) )
        W = W1
        if lim < 0.0001:
            break
    return W
```

２．ネゲントロピー最大化に基づいての独立成分分析

もとのデータでは

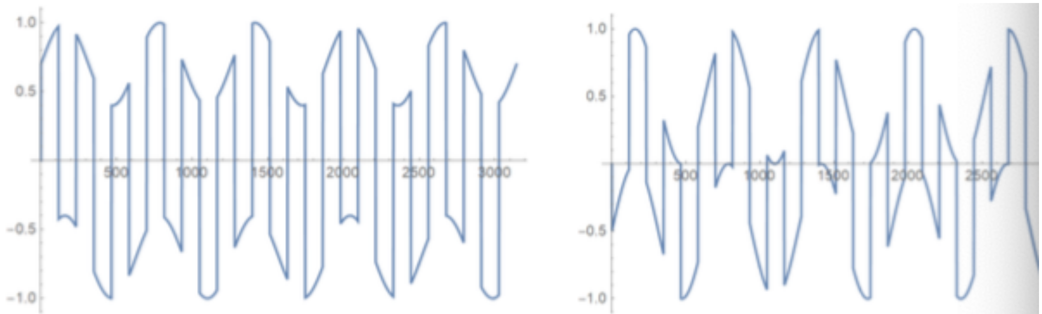
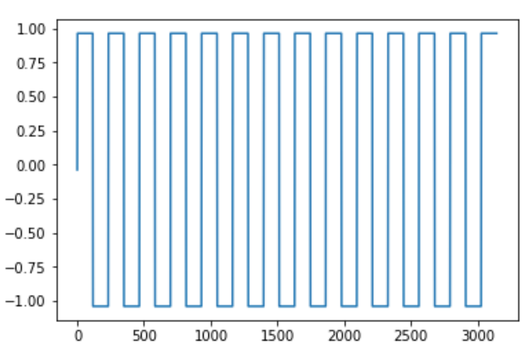
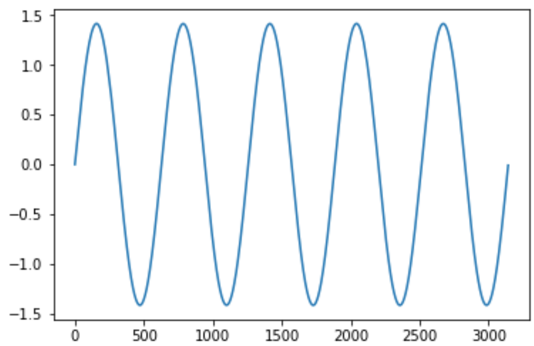


図1 ：テスト用のデータ．左から dat1.txt, dat2.txt.

処理したデータでは





結果は同じである。最初wはランダムで選択するので、かかった時間の比較は必要なしと思う。

IV. 参考文献

https://blog.csdn.net/zb1165048017/article/details/48464573

My gitlab: https://doss-gitlab.eidos.ic.i.u-tokyo.ac.jp/ddwhzh/ica