独立成分分析 (ICA) 与其在音频信号分离中的应用

张桢锴

2025.5.3

摘要

本文简要介绍了独立成分分析（ICA）特别是 FastICA 算法的基本原理与实现流程。ICA 利用信号的非高斯性与统计独立性，从多个观测信号中恢复原始独立源信号。文中详细阐述了数学建模、信号预处理（中心化与白化）、固定点迭代过程，并通过代码与实验演示了 FastICA 在模拟音频信号分离中的实际效果，显示其在语音分离与降噪等的应用。

关键词：独立成分分析；FastICA 算法；音频处理；噪音消除

1 前置知识

高斯分布与非高斯分布

ICA 依赖源信号的非高斯性（Non-Gaussianity）作为分离的依据

高斯信号指信号的幅值分布服从高斯分布（正态分布），**非高斯信号**指幅值分布不服从高斯分布，可能呈现双峰、尖峰、重尾或不对称性（如语音、图像边缘、脑电信号）. 而我们关注的恰恰是这些具有不对称性的非高斯信号.

由中心极限定理，多个独立随机变量的混合信号趋于服从高斯分布，因此通过最大化分离信号的非高斯性，能分离处出原始的信号.

信号的统计独立性

把信号 $s_1(t), s_2(t)$ 视为随机变量，则其相互独立为 $p(s_1, s_2) = p(s_1)p(s_2)$.

同理 n 个信号相互独立即为 $p(s_1, s_2, \cdots, s_n) = \prod_{i=1}^{n} p(s_i)$.

2 独立成分分析 (ICA) 的基本原理

在现实生活中，观测信号（例如麦克风接收的声音信号）$\boldsymbol{X}$ 为源信号（例如各个环境中的声源）$\boldsymbol{S}$ 的瞬时线性组合，且源信号可以认为相互统计独立。我们试图从观测信号中提取出各个相互独立的源信号，并求出其混合矩阵 $\boldsymbol{A}$.

2.1 数学模型

设观测信号由 n 个相互统计独立的源信号线性组合而成，数学形式可表述为：

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$$
$$\boldsymbol{s} = (s_1(t), s_2(t), \cdots, s_n(t))^T$$

其中：

- 信号源 $s_i(t)$ 表示第 i 个信号在时间 t 的实际值。其中 $p(s_1, s_2, \cdots, s_n) = \prod_{i=1}^{n} p(s_i)$, 即 s_i 相互独立.
- $\boldsymbol{x}$ 为观测信号向量，包含 m 个通道在时间 t 的采样值.
- $\boldsymbol{s}$ 为独立源信号向量，即 n 个信号源在 T 个时间点的实际值.
- $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ 为未知的混合矩阵，表征原信号到观测信号的线性映射关系。一般假设 $n \le m$, 以保证模型可辨识性.

对于 T 个时间点的采样值，观测信号作为列向量构成观测信号矩阵 $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T)$, 同理源信号构成源信号矩阵 $\boldsymbol{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_T)$, 同样有

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$$

2.2 ICA 的目标

这里我们不加证明的断言，在 ICA 的假设中，信号的**非高斯性**等价于信号之间的**独立性**.

为尽可能还原源信号，我们的目标是找到解混矩阵 $\boldsymbol{W}$, 使分离出的信号 $\boldsymbol{Y} = \boldsymbol{W}^T\boldsymbol{X}$ 尽可能的接近源信号 $\boldsymbol{S}$. 也即对 $\boldsymbol{W}$ 的每个列向量 $\boldsymbol{w}_i$，投影结果 $\boldsymbol{y}_i = \boldsymbol{w}_i^T\boldsymbol{X}$ 尽量满足非高斯性.

对于非线性函数 $G(x)$ (例如 $\log_2 \cosh x$，用于放大非高斯特征)，最大化负熵近似 $J(y) = (E(G(y)) - E(G(v)))^2$ 能够使 y 的非高斯性最大. 其中 v 为满足标准正态分布的随机变量.

3 FastICA 算法

FastICA 是一种基于固定点迭代的独立成分分析（ICA）算法. 其核心思想是通过最大化非高斯性来估计独立成分.

3.1 预处理

预处理的目的是将观测信号各个行（每个观测点不同时间采集的样本）的均值变为 0, 方差变为 1.

中心化

将观测信号 $\boldsymbol{X} \in \mathbb{R}^{n\times T}$ 零均值化，即

$$\boldsymbol{X}_{\mathrm{c}} = \boldsymbol{X} - \mathbb{E}(\boldsymbol{X})$$

其中 $\mathbb{E}(\boldsymbol{X})$ 为 $\boldsymbol{X}$ 每个行的均值组成的列向量.

白化

对中心化后的数据进行白化，使数据矩阵正交.

1. 计算协方差矩阵

$$\boldsymbol{C} = T \cdot \mathrm{Cov}(\boldsymbol{X}_{\mathrm{c}}) = \boldsymbol{X}_{\mathrm{c}}\boldsymbol{X}_{\mathrm{c}}^{T}$$

如果 $\boldsymbol{C}$ 半正定但不正定，则表明 $\boldsymbol{X}_{\mathrm{c}}$ 中有线性相关的维度，除去冗余维度即可.

若 $\boldsymbol{C}$ 正定，则其特征值均大于零.

2. 对 $\boldsymbol{C}$ 相似对角化，即

$$\boldsymbol{C} = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^{T}$$

考虑 $\boldsymbol{C}$ 正定的情况，$\boldsymbol{E}$ 为单位正交向量矩阵，$\boldsymbol{\Lambda}$ 为特征值对角阵.

3. 构造白化矩阵，即

$$\boldsymbol{W}_{\mathrm{whiten}} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{E}^{T}$$

4. 得到白化后的数据

$$\tilde{\boldsymbol{X}} = \boldsymbol{W}_{\mathrm{whiten}}\boldsymbol{X}_{\mathrm{c}}$$

此时 $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^{T} = \boldsymbol{X}_{\mathrm{c}}^{T}\boldsymbol{E}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{E}^{T}\boldsymbol{X}_{\mathrm{c}} = \boldsymbol{I}$, 即 $\tilde{\boldsymbol{X}}$ 是正交阵.

3.2 固定点迭代提取独立成分

我们要提取 n 个独立成分，即分离矩阵 $\boldsymbol{W} = \begin{pmatrix} \boldsymbol{\omega}_1^T \\ \vdots \\ \boldsymbol{\omega}_n^n \end{pmatrix}$ 使得 $\boldsymbol{Y} = \boldsymbol{W}\tilde{\boldsymbol{X}}$ 为目标原信号矩阵的近似.

对 $\tilde{\boldsymbol{X}}$ 的某个列向量 $\boldsymbol{x}$，我们要最大化 $J_G(\boldsymbol{\omega}^T\boldsymbol{x}) = \big(E(G(\boldsymbol{\omega}^T\boldsymbol{x})) - E(G(v))\big)^2$. 限定 $\|\boldsymbol{\omega}\| = 1$，为使用 Lagrange 乘数法，对 $\boldsymbol{\omega}$ 求偏导得

$$\frac{\partial\left(J_G(\boldsymbol{\omega}^T\boldsymbol{x}) - \lambda\left(\|\boldsymbol{\omega}\|^2 - 1\right)\right)}{\partial\boldsymbol{\omega}} = 2\mathbb{E}\left(\boldsymbol{x}G'(\boldsymbol{\omega}^{\boldsymbol{T}}\boldsymbol{x})\right) - 2\lambda\boldsymbol{\omega}$$

使上式为 0 的 $\boldsymbol{\omega}$ 即为最优的 $\boldsymbol{\omega}$. 使用牛顿法 $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$, 能得到上述方程的近似解. 计算偏导

$$\frac{\partial\left(\mathbb{E}\left(\boldsymbol{x}G'(\boldsymbol{\omega}^{\boldsymbol{T}}\boldsymbol{x})\right) - \lambda\boldsymbol{\omega}\right)}{\partial\boldsymbol{\omega}} = \mathbb{E}\left(\boldsymbol{x}\boldsymbol{x}^TG''\left(\boldsymbol{\omega}^{\boldsymbol{T}}\boldsymbol{x}\right)\right) - \lambda\boldsymbol{I} \approx \mathbb{E}\left(G''\left(\boldsymbol{\omega}^T\boldsymbol{x}\right)\right)\boldsymbol{I} - \lambda\boldsymbol{I}$$

代入牛顿法公式，用 $\boldsymbol{\omega}_n$ 近似 λ，并进一步化简最后得

$$\boldsymbol{\omega}_{n+1} = \mathbb{E}\left(\boldsymbol{x}G'\left(\boldsymbol{\omega}_n^T\boldsymbol{x}\right)\right) - \mathbb{E}\left(G''(\boldsymbol{\omega}_n^T\boldsymbol{x})\right)\boldsymbol{\omega}_n$$

对多个 $\boldsymbol{x}$ 的计算方法类似. 算法的具体过程如下：

1. 随机生成正交阵 $\boldsymbol{W}^{(0)} \in \mathbb{R}^{n\times n}$. 一般采用随机生成矩阵结合 QR 分解的方法实现.

2. 固定点迭代更新.

$$\boldsymbol{W}^{(n+1)'} = \mathbb{E}\left[\tilde{\boldsymbol{X}}G'\left(\boldsymbol{W}^{(n)T}\tilde{\boldsymbol{X}}\right)\right] - \mathbb{E}\left[G''\left(\boldsymbol{W}^{(n)T}\tilde{\boldsymbol{X}}\right)\right]\boldsymbol{W}^{(n)}$$

$$\boldsymbol{W}^{(n+1)} = \left(\boldsymbol{W}^{(n+1)'}\boldsymbol{W}^{(n+1)'T}\right)^{-1/2}\boldsymbol{W}^{(n+1)'}$$

第二个式子的归一化与白化过程基本相同，保持了 $\boldsymbol{W}$ 的正交性.

3. 不断迭代直至收敛. 例如 $\Delta = \left\| \boldsymbol{W}^{(n+1)} - \boldsymbol{W}^{(n)} \right\|_F$ 小于特定值.

此时的 $\boldsymbol{W}$ 即为所求解混矩阵.

4 FastICA 算法对模拟音频信号的分离

首先使用代码进行正弦波、方波与锯齿波的生成，并随机生成一些噪声模拟声音信号叠加到原波形上. FastICA 处理的 Python 代码如下（部分）

```python
def g(x): return np.tanh(x)
def g_prime(x): return 1 - np.tanh(x) ** 2

def whiten(X):
    X -= X.mean(axis=0)
    cov = np.cov(X, rowvar=False)
    d, E = np.linalg.eigh(cov)
    D_inv = np.diag(1. / np.sqrt(d))
    return X @ E @ D_inv @ E.T

def fastica(X, n_components, max_iter=200, tol=1e-4):
    X = whiten(X)
    n_samples, n_features = X.shape
    W = np.zeros((n_components, n_features))

    for i in range(n_components):
        w = np.random.rand(n_features)
        w /= np.linalg.norm(w)
        for _ in range(max_iter):
            wx = X @ w
            gwx = g(wx)
            g_wx = g_prime(wx)
            w_new = (X.T @ gwx - g_wx.mean() * w) / n_samples
            for j in range(i):
                w_new -= np.dot(w_new, W[j]) * W[j]
            w_new /= np.linalg.norm(w_new)
            if np.abs(np.abs(np.dot(w_new, w)) - 1) < tol:
                break
            w = w_new
        W[i, :] = w
    return X @ W.T

S_est = fastica(X, n_components=3)
```

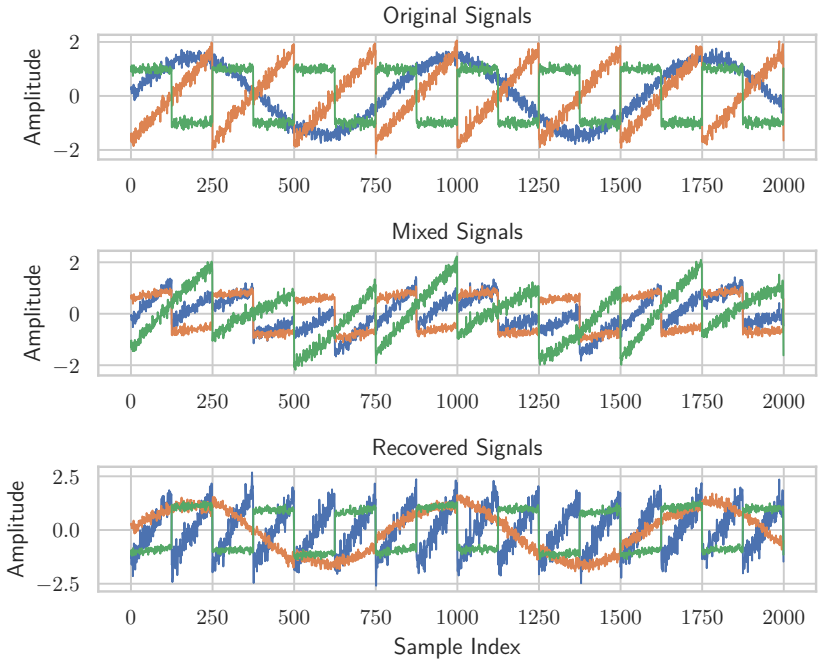

图 1: FastICA recovered signals

可视化后所得结果如下图. 可见该算法能清晰得分辨三种信号并将其分离，对原信号的振幅、频率、波形等参数做了很好的还原.

对于真实的 `.wav` 格式的文件，可以使用 Python 的 `scipy.io.wavfile` 库函数进行读入并进行类似的处理，能够分离出一段音频中的人声和其它声音，也能分离出男声和女声. 同时，若将音频中提取出的环境噪音消去，则能实现降噪的效果.

5 遇到的问题与展望

- **遇到的问题与参考说明** 在查阅不同资料（包括网站博客与相关论文原文）时，发现不同文章中白化方式、迭代处理等方面存在一定的差异，对某些矩阵的定义有一定出入。文[1]中分别介绍了单点迭代和多点迭代两种算法的不同与相应继承关系，定义较为明确，故采用之。应用场景参考了文[3], 进行了盲源音频分离的模拟.

  在 AI 使用上，本文在信号生成与可视化上借助了 ChatGPT-o4 进行生成，减小了代码工作量。

- **展望与改进点**

  1. 在信号采样率很高时，矩阵的维度会很大，上述算法中大量的矩阵分解等运算计算复杂度较高. 可以使用随机 SVD 降维等方式降低计算量.
  2. ICA 假设信号为瞬时线性混合，但是现实中由于不同信号的传播路径不同，接受到信号具有**时间延迟**和**滤波效应**. 此时借鉴文[2]采用卷积来混合源信号，能够消除上述干扰.
  3. ICA 严格依赖原信号的非高斯性，对亚高斯信号分离能力差，需要使用更复杂的算法实现对信号的处理.

4. 音频处理不仅时基于现有数据的静态数据，也有需要实时处理的动态数据例如降噪耳机的算法，需要对该算法进行一些扩展.

- **本人的一些新发现**

1. 发现基于特征值分解的白化方法可以代替施密特正交化来将一个矩阵在保持列空间不变的条件下转化为正交阵，改方法计算远远方便于施密特正交化.
2. 发现在矩阵、向量的多元函数中仍然能够使用牛顿法获得方程的近似解.

参考文献

[1] HYVÄRINEN A, OJA E. Independent component analysis: algorithms and applications[J]. Neural networks, Elsevier, 2000, 13(4-5): 411-430.

[2] SW_ 孙维. 盲源分离（*BSS*）的学习总结（*PCA*、*ICA*）[EB/OL]. CSDN 博客, 2025. https://wenku.csdn.net/column/3tjk93vie4.

[3] hh867308122. 【*ICA* 独立成分分析】数学原理 *+python* 代码实现 [EB/OL]. CSDN 博客, 2024. https://blog.csdn.net/hh867308122/article/details/144175594.