Separation of Independent Audio Source from Mixture

Department of Automatic Control & Decision Systems

Authors:
Erkid Hoxholli
Nandan Hegde

Appendix

1. The problem statement

Let’s suppose we record two or more signals obtained from two or more independent speech sources and we mix them. After that we try to perform blind source separation using FastICA to extract two or more source signals from the mixture.

In our case we used three signals, mixed them using different mixing coefficients and extracted the source signals using the FastICA implementation as described in the following pages.

In our implementation we have the following signals:

- src1.wav: Ellen Degeneres speech
- src2.wav: Mark Zuckerberg speech
- src3.wav: Anne Hathaway speech

In the next chapter we are going to have a look at the actual implementation details explained in a mathematical notation, as well as look at the source code built using Matlab.

2. What is FastICA

FastICA is an algorithm used for independent component analysis (ICA). It is a really efficient and simple algorithm used a lot for blind source separation problems (BSS). In our case we are using FastICA to separate a set of source signals from a mixed signal with the need of very little information about the source and the process used to mix the sources. The source signals in our case are audio signals in a wav format.

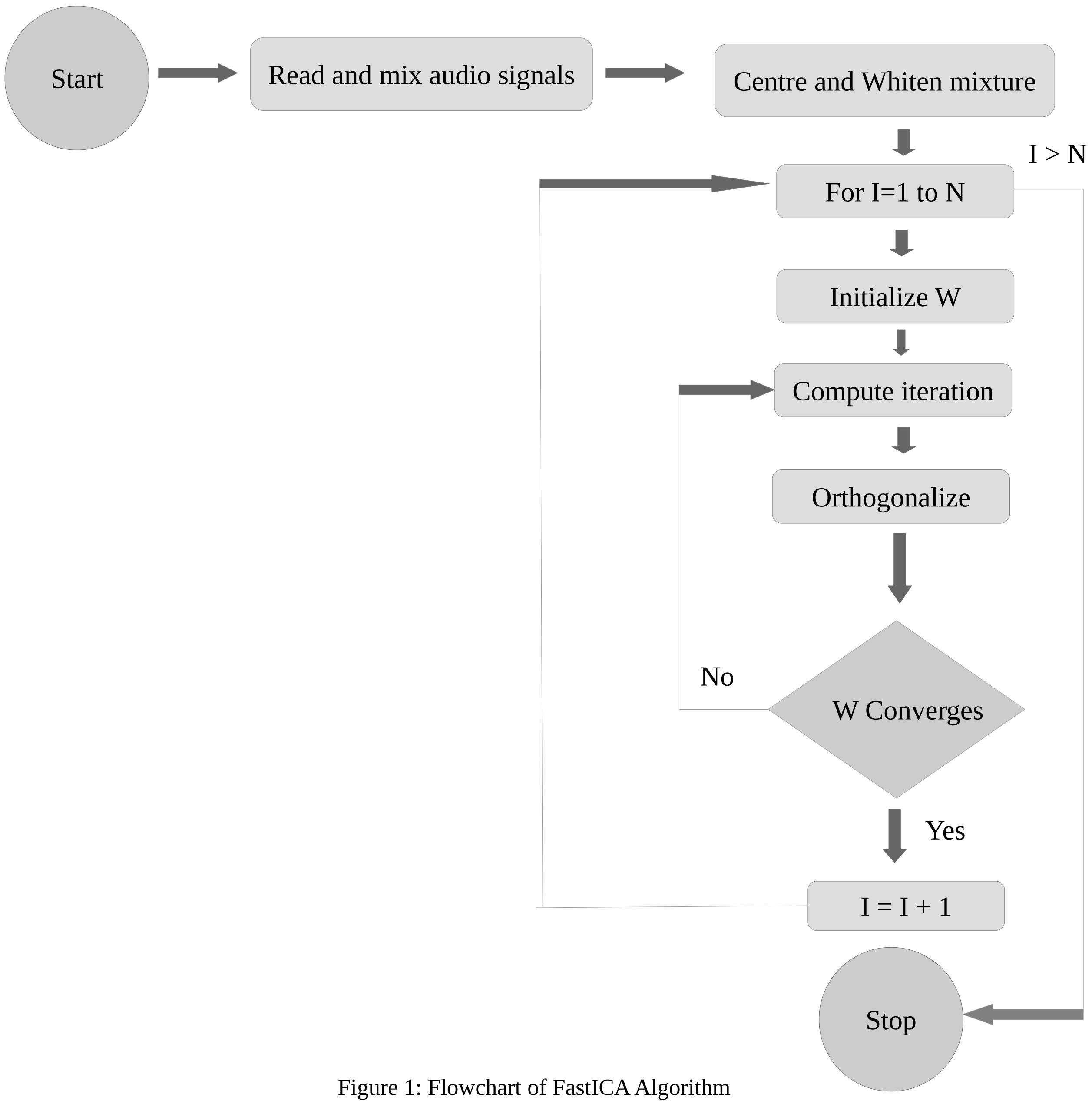

Figure 1: Flowchart of FastICA Algorithm

3. FastICA Implementation

In the chapter we are going to have a look at the actual implementation of FastICA, starting with reading sources, creating the mixture using mixing coefficients, preprocessing our mixture as well as writing the extracted source back into the local folder after applying FastICA.

3.1 Reading sources and creating our mixture

N → number of samples
S_1 → first source
$S_1 = S_1(1), S_1(2) \dots S_1(N)$
S_2 → **second source**
$S_2 = S_2(1), S_2(2) \dots S_2(N)$
S_3 → **third source**
$S_3 = S_3(1), S_3(2) \dots S_3(N)$

Now let’s create three mixtures using the sources above:
where **C_1, C_2, C_3** are our mixing coefficients
We create the mixtures the following way, by multiplying our sources with a specific mixing coefficient.

$M1 = \begin{bmatrix} S_1 C_1 \\ S_2 C_2 \\ S_3 C_3 \end{bmatrix} M2 = \begin{bmatrix} S_1 C_2 \\ S_2 C_1 \\ S_3 C_3 \end{bmatrix} M3 = \begin{bmatrix} S_1 C_3 \\ S_2 C_1 \\ S_3 C_2 \end{bmatrix}$, where M1, M2, M3 are three mixtures

Final linear mixture:
M = M1 + M2 + M3

3.2 Preprocessing: Centering and Whitening

Centering

First we estimate the mean value for all of our sources:

$$\widehat{m_{S1}}=\frac{1}{N}\sum_{i=1}^{N}S_1(i)$$

$$\widehat{m_{S2}}=\frac{1}{N}\sum_{i=1}^{N}S_2(i)$$

$$\widehat{m_{S3}}=\frac{1}{N}\sum_{i=1}^{N}S_3(i)$$

Secondly we calculate the centered sequence:

$$\widetilde{S_1}(i)=S_1(i)-\widehat{m_{Si}}$$

$$\widetilde{S_2}(i)=S_2(i)-\widehat{m_{Si}}$$

$$\widetilde{S_3}(i)=S_3(i)-\widehat{m_{Si}}$$

$$\widetilde{S}=[\widetilde{S_1}\ldots\widetilde{S_i}]$$

Whitening**:**

A whitening transformation is a linear transformation that transforms a vector of random variables with a known covariance matrix into a set of new variables whose covariance is the identity matrix, meaning that they are uncorrelated and each have variance 1. The transformation is called "whitening" because it changes the input vector into a white noise vector.

1.Let x be a vector of zero-mean data. Form its covariance matrix

$$\Sigma=E(xx^T)$$

where the operator E denotes the expected value (mean) of its argument.

$$E(x(t))=\frac{1}{N}\sum_{t=1}^{N}x(t)$$

If the data points in x are correlated, then their covariance, Σ, will NOT be a diagonal matrix.

2.*In order to decorrelate the data, we need to transform it so that the transformed data will have a diagonal covariance matrix. This transform can be found by solving the eigenvalue problem. We find the eigenvectors and associated eigenvalues of Σ by solving*

$$\Sigma\Phi=\Phi\Lambda$$

Λ is a diagonal matrix having the eigenvalues as its diagonal elements. The matrix Φ thus diagonalizes the covariance matrix of x. The columns of Φ are the eigenvectors of the covariance matrix. We can also write the diagonalized covariance as:

$$\Phi^T\Sigma\Phi=\Lambda \quad \textbf{(1)}$$

If we wish to apply this diagonalizing transform to a single vector of data we just form:

$y=\Phi^T x$ **(2)**

Thus, the data y has been decorrelated: its covariance, $E[yy^T]$ is now a diagonal matrix, Λ

3.The diagonal elements (eigenvalues) in Λ may be the same or different. If we make them all the same, then this is called whitening the data. Since each eigenvalue determines the length of its associated eigenvector, the covariance will correspond to an ellipse when the data is not whitened, and to a sphere (having all dimensions the same length, or uniform) when the data is whitened. Whitening is easy:

$I=\Lambda^{\frac{-1}{2}}\Lambda\Lambda^{\frac{-1}{2}}$

Equivalently, substituting in (1), we write:

$I=\Lambda^{\frac{-1}{2}}\Phi^T\Sigma\Lambda^{\frac{-1}{2}}$

Thus, to apply this whitening transform to y we simply multiply it by this scale factor, obtaining the whitened data w:

$W=\Lambda^{\frac{-1}{2}}y=\Lambda^{\frac{-1}{2}}\Phi^{Tx}$ **(3)**

Now the covariance of W is not only diagonal, but also uniform (white), since the covariance of w,

$I=E(WW^T)$

$I=E(\Lambda^{\frac{-1}{2}}\Phi^T xx^T\Phi\Lambda^{\frac{-1}{2}})$

In DHS, the diagonalizing transform applied to x is denoted A and the whitening transform is represented by Aw. These map onto the notation above as follows

$y=A^{Tx}, A=\Phi$

$w=A_W^{Tx}, A_W=\Phi\Lambda^{\frac{-1}{2}}$

3.3 FastICA implementation

Iteration algorithm:

Initialize $W=\begin{bmatrix}1,1,1\\1,1,1\\1,1,1\end{bmatrix}$

Initialize random vectors w0, w1 with size 3x1:

$W0=[W0_{1,1},W0_{2,1},W0_{3,1}]^T$

$W1=[W1_{1,1},W1_{2,1},W1_{3,1}]^T$

Perform one source extraction algorithm:

$\widetilde{W}_i=avg\,\Phi(t)[W^T{}_{i=1}\Phi(t)]^3-3W_{i-1}$, where $\Phi(t)=[\Phi_1(t),\Phi_2(t),\Phi_3(t)]^T$

$If\ W^T{}_i W_{i-1}\ is\ not\ close\ enough$ to 1 $we\ start\ the\ iteration\ algorithm\ again$

Orthogonalization:

$\psi=\sum_{j=1}^{N-1} W1^T*W_j*W_j\,,\,where\,W_j=[W_{1xj},W_{2xj},W_{3xj}]^T$, where ψ is the orthogonality vector

$W1=W1-\psi$ and we normalize W1

In order to get back the original source signals we use the following

$\underline{S}=W^T*\Phi$ where $\underline{S}$ is the matrix with the extracted components

4. Results

To sum up we would like to show the resulted extracted audio using FastICA by representing their plots, which is easier to visualize the results of the algorithm.
We can see from by comparing original and extracted plots that the results are acceptable considering mixing coefficients, simplicity of the algorithms and using three different sources.

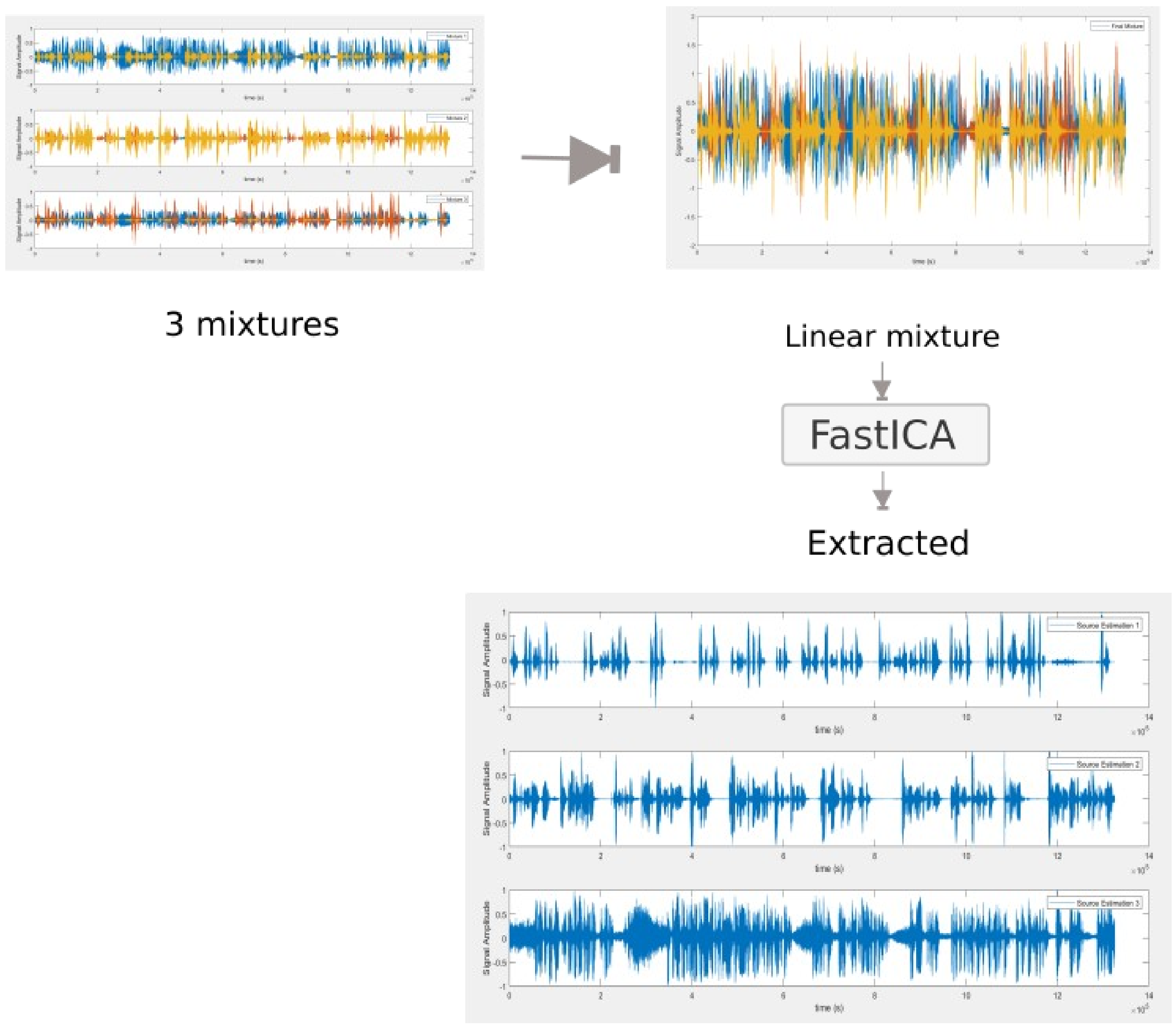

Figure 2: Plots representing 3 mixtures, the linear mixture and extracted sources using FastICA algorithm

5. Code

```
C1 = 1;
C2 = 0.1;
C3 = 0.5;

%% Generating Mixture

Amix = rand(N,N);

% Reading the audio
S1 = audioread('Src11.wav');
S2 = audioread('Src22.wav');
S3 = audioread('Src44.wav');

S1(1325601:end,:) = [];
S2(1325601:end,:) = [];
S3(1325601:end,:) = [];

mix1 = [S1' * C1; S2' * C2; S3' * C3];
mix2 = [S1' * C2; S2' * C3; S3' * C1];
mix3 = [S1' * C3; S2' * C1; S3' * C2];

audiowrite('Mix1.wav',mix1',44100); %may be M instead of Xobs
audiowrite('Mix2.wav',mix2',44100); %may be M instead of Xobs
audiowrite('Mix3.wav',mix3',44100); %may be M instead of Xobs

% Linear mixture of the independent audio source
M = mix1 + mix2 + mix3 ;

X_observe = Amix*M;                             %Matrix consisting of M samples of N observed mixtures

%% Preprocessing, Centering
X = X_observe';

Xmean = mean(X);

for i = 1:N
    X(:,i) = X(:,i) - Xmean(i);
end

%% Preprocessing, Whitening

Es    = cov(X);
[Eigen_val,Eigen_vector] = eig(Es);

% whitened matrix
Z = Eigen_val*1/sqrt(Eigen_vector)*Eigen_val'*X';

%% FastICA algorithm

W = ones(N,N);
for p = 1:N

    %Initiaize row matrix w0 and w1
    w1 = randn(N, 1);
    w1 = w1/norm(w1,2);
    w0 = randn(N, 1);
    w0 = w0/norm(w0, 2);

    while (abs(abs(w0'*w1)-1) > 0.001)
        w0 = w1;
        w1 = mean(Z.*power(w1'*Z, 3), 2) - 3*w1;
        w1 = w1/norm(w1, 2);
    end

    orth  = zeros(N,1);

    for j = 1:p-1
        orth = orth + w1'*W(:,j)*W(:,j);
    end

    w1      = w1 - orth;
    w1      = w1 / sqrt(w1'*w1);
    W(:,p) = w1;
end
Sep = W'*Z;

%% Plotting and saving seperated audios

figure
subplot(311)
plot(Sep(1,:))
xlabel('time (s)')
ylabel('Signal Amplitude')
legend('Source Estimation 1')

subplot(312)
plot(Sep(2,:))
xlabel('time (s)')
ylabel('Signal Amplitude')
legend('Source Estimation 2')

subplot(313)
plot(Sep(3,:))
xlabel('time (s)')
ylabel('Signal Amplitude')
legend('Source Estimation 3')

audiowrite('Sep_1.wav',Sep(1,:),44100);
audiowrite('Sep_2.wav',Sep(2,:),44100);
audiowrite('Sep_3.wav',Sep(3,:),44100);
```

6. References

DELORME, A. (n.d.). ICA for dummies. Retrieved from http://arnauddelorme.com/ica_for_dummies/

Eigenvalues and Eigenvectors. (n.d.). Retrieved from https://www.math.hmc.edu/calculus/tutorials/eigenstuff/