Idea about ICA

Consider an example of “Cocktail Party Problem” where there are ‘d’ speakers speaking simultaneously at a cocktail party and few microphones placed in the room are recording voices. Because each microphone is at a different distance from each of the speakers, it records a different combination of the speakers’ voices. Assume we have ‘d’ such microphones and thus, ‘d’ observations, so our task is to recover the original ‘d’ speaker’s voices from these observations…

For this, we assume that the data(mixed signals) $x \in R^d$ is generated by $'d'$ independent sources. Thus, our goal is to recover the sources $'s'$ used to generate our mixed data(in this example, mixed sounds).
For simplicity, assume the source signals ‘s’ were mixed in a linear sense. So, observed data ‘x’ can be represented by:

$$x(i) = As(i)\,, where\ A - mixing\ matrix$$
$$\&\ s(i)\ \&\ x(i)\ are\ vectors(signals)\colon i = 1,2,\ldots,d$$

So, for calculating the sources $'s'$, we now need to somehow find the unmixing matrix W such that,

$$s(i) = Wx(i)\,, where\ W - unmixing\ matrix\ i.e.\ W^{-1} = A$$

So, our algorithm goes as follows….

ICA Algorithm (Theory):

Assume that input data is given as a matrix $X \in R^{N \times M}$ where M, the number of columns indicate the number of samples of mixed signals and N, the number of rows corresponding with the number of source signals.

- **Preprocessing the data:**

1) **Center the data** - Here, we basically subtract the mean of columns of data from each column of X. (Usually done to simplify the ICA algorithm.)

$$X(i) \leftarrow X(i) - \bar{x} \text{ , for each column 'i' of } X \text{ .}$$

& $\bar{x}$ *is a mean vector of columns of* X

2) **Whitening** - A linear transformation that un-correlates the data points and makes the covariance matrix of transformed data to be an identity matrix. (It reduces the complexity of ICA algorithm by reducing the number of parameters to estimate from N^2 *to* $N(N-1)/2$)

   Covariance matrix, $E[XX^T]$, being real-symmetric, can be decomposed using an eigenvalue decomposition.
   So,

$$E[XX^T\} = EDE^T$$

Then after whitening, ‘X’ becomes:

$$X \leftarrow D^{-1/2}E^T X$$

- Moreover, in this step, as we are already decomposing covariance matrix using eigenvalue-decomposition, we may use PCA for dimensionality reduction which will make the data less complex by removing unwanted components.

- **Component Extraction:**

In this particular algorithm, the Central Limit theorem is used for deriving steps. According to CLT, the sum of independent random variables tends towards a Gaussian Distribution. So, the observed signals must have more Gaussianity(extent to which something is gaussian) than the original signals i.e. they tend to exhibit more Gaussian behavior than Original signals. Now, to check for Gaussianity, there is a measure known as **Negentropy** which is simply obtained by subtracting entropy of a given random variable(y) from entropy of Gaussian Random variable(v) which has same covariance matrix and mean as that of given rv. As entropy is always maximum for Gaussian rv(v), this measure is always going to be non-negative and zero only when the given rv is gaussian. Also, this measure is statistically more well defined than other measures of Gaussianity like Kurtosis.

But, it is hard to estimate Negentropy from the theoretical formula. So, we have to look for some better approximations. Hyvärinen gave some approximations by conic combinations of non-quadratic functions which were more useful than other approximations. Also, even if we take any one of those non-quadratic functions, it can still be considered as a measure of non-gaussianity. Moreover, by choosing a non-quadratic function which does not grow too fast, we can get robust and computationally easy-to-estimate measure of non-gaussianity.

So, in this FastICA algorithm, the functions that are being used are:

$$f_1(u) = log(cosh(u))$$

$$f_2(u) = -e^{-u^2/2}$$

And then our approximate measure becomes:

$$J(y)\ \alpha\ [E\{f(y)\} - E\{f(v)\}]^2\ where\ f\ is\ either\ f_1\ or\ f_2.$$

Note that this approximation assumes that rv ‘y’ and normal rv ‘v’ has unit variance and zero mean which is done by preprocessing of data.

FastICA Algorithm

This algorithm is simply a gradient descent algorithm which finds rows of matrix 'W' that maximizes the objective function given above.

- **For single component:**
  For finding single column of W^T(denoted by 'w'), the optimization problem will be:

$$max \ [E\{f(w^T X)\} - E\{f(v)\}]^2$$
$$s.t. \ ||w|| = 1$$

  Note that the constraint applied here is because we have to make sure that variance of 'y' should be 1.(Here, y = $w^T X$)
  Then by applying KKT conditions, and solving the newly formed problem by Newton's method, we arrive at following algorithm:

1. Choose an initial (e.g. random) weight vector $\mathbf{w}$.
2. Let $\mathbf{w}^+ = E\{\mathbf{x}g(\mathbf{w}^T\mathbf{x})\} - E\{g'(\mathbf{w}^T\mathbf{x})\}\mathbf{w}$
3. Let $\mathbf{w} = \mathbf{w}^+ / \|\mathbf{w}^+\|$
4. If not converged, go back to 2.

  Where g(u) is the first derivative of f(u).

- **For several components:**
  The above algorithm will give you one row of matrix ‘W’ and thus, one independent component out of all possible components.
  Thus, to get all the components, we have to extend this algorithm as follows:

Algorithm FastICA

Input: C Number of desired components

Input: $\mathbf{X} \in \mathbb{R}^{N \times M}$ Prewhitened matrix, where each column represents an N-dimensional sample, where $C <= N$

Output: $\mathbf{W} \in \mathbb{R}^{N \times C}$ Un-mixing matrix where each column projects $\mathbf{X}$ onto independent component.

Output: $\mathbf{S} \in \mathbb{R}^{C \times M}$ Independent components matrix, with M columns representing a sample with C dimensions.

for p **in** 1 to C:

$\mathbf{w_p} \leftarrow$ *Random vector of length N*

while $\mathbf{w_p}$ changes

$$\mathbf{w_p} \leftarrow \frac{1}{M}\mathbf{X}g(\mathbf{w_p}^T\mathbf{X})^T - \frac{1}{M}g'(\mathbf{w_p}^T\mathbf{X})\mathbf{1_M}\mathbf{w_p}$$

$$\mathbf{w_p} \leftarrow \mathbf{w_p} - \sum_{j=1}^{p-1}(\mathbf{w_p}^T\mathbf{w_j})\mathbf{w_j}$$

$$\mathbf{w_p} \leftarrow \frac{\mathbf{w_p}}{\|\mathbf{w_p}\|}$$

output $\mathbf{W} \leftarrow [\mathbf{w_1}, \ldots, \mathbf{w_C}]$

output $\mathbf{S} \leftarrow \mathbf{W^T X}$

After running this algorithm, we have to again multiply by inverse of the whitening matrix and add the mean to the data, in order to get the complete recovered signals.

Implementation and Conclusions

We applied this FastICA algorithm for two cases:

1) Separating Voices (Cocktail Party Problem) and
2) Separating Images from Mixed Images

Based on these implementations, we conclude that:

1) The FastICA algorithm works well in both of the above-mentioned cases.
2) But Independence assumption plays a key role in this algorithm as if original sources are not sufficiently independent, then this algorithm fails.
3) This algorithm works well even in cases when there is some noise in the data. As, the centering part removes that noise (or at least reduces it to a greater extent).

References:

https://en.wikipedia.org/wiki/Independent_component_analysis#:~:text=In%20signal%20processing%2C%20independent%20component,statistically%20independent%20from%20each%20other.

https://en.wikipedia.org/wiki/FastICA

https://www.cs.helsinki.fi/u/ahyvarin/papers/NN00new.pdf

https://github.com/shantistewart/Independent-Component-Analysis/blob/master/Paper/Research%20Paper.pdf