Independent Component Analysis

Nicolas Brissonneau , UTEID:nb24488 , email:nicolasb@utexas.edu

I. Introduction

I am considering the problem of recovering n independently produced soundtracks which have been artificially mixed together randomly. In order to achieve this, I will make use of Independent Component Analysis (ICA) by maximizing the entropy of the system through a gradient descent approach.

II. Method

Let's consider the n by t matrix U containing the n soundtrack samples of duration t, and a "mixing" m by n matrix A left-multiplying U to obtain the m by t matrix X:

$$X = AU \tag{1}$$

The goal is now to recover an approximation Y of U by finding the n by m matrix W such that:

$$Y = WX \tag{2}$$

Using the gradient descent approach, we are looking for a cost function which will "reward" the signals recovery matrix W when it is updated and becomes closer to "unmixing" the X matrix. The independent component analysis relies on the assumption that the original soundtracks are not correlated to each other, so if a correlation is found between two of the mixed signals then it means that at least one soundtrack is being correlated to itself.

A. Gradient descent

To design the cost function, we rely on the assumption of the n source signals s being independent:

$$p(s) = \prod_{i=1}^{n} p_s(s_i) \tag{3}$$

In our case we want the mixed signal pdf:

$$p(x) = \prod_{i=1}^{n} p_s(\omega_i^T x) \cdot |W| \tag{4}$$

But because we don't know it, we rely on the cdf approximation and its log likelihood:

$$g(s) = \frac{1}{1+e^{-s}} \tag{5}$$

$$l(W) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \log g'(\omega_j^T x^{(i)}) + \log |W| \right) \tag{6}$$

Finally, differentiating this expression with respect to W leads us to the cost function J:

$$J = \begin{bmatrix} 1-2g(\omega_1^T x^{(i)}) \\ \vdots \\ 1-2g(\omega_n^T x^{(i)}) \end{bmatrix} x^{(i)^T} + (W^T)^{-1} \tag{7}$$

For computational convenience, we are right-multiplying J by $W^T W$ and add the scalar product associated with the learning rate α:

$$\Delta W = \alpha(I + (1-2Z)Y^T)W \tag{8}$$

Where ΔW is an increment, we can now make W converge by redefining it over each iteration as:

$$W_{i+1} := W_i + \Delta W \tag{9}$$

B. Learning rate varying over iterations

We know that the learning rate has an enormous impact on the convergence. If it is set too high, we will never get close to the minimum we are looking for, if it is too low we can get there, but over a considerable amount of iterations. I am experimenting with learning parameter variations over iterations, trying to converge faster and more accurately than by using a constant value. I have explored different approaches:

$$\alpha_{smart} = cst \tag{10}$$

$$\alpha_{smart} = \frac{\alpha_{max} + \alpha_{min} i}{i+1} \tag{11}$$

$$\alpha_{smart} = (\alpha_{max} - \alpha_{min}) \frac{N-i}{N} + \alpha_{min} \tag{12}$$

Where $[\alpha_{min}, \alpha_{max}]$ represent the learning rate boundaries, i the iteration number and N the total number of iterations planned.

III. Results

A. Main recovery result

An example of the results of this algorithm follows below with Figure 1, Figure 2 and Figure 3. We observe in Figure 1 the initial soundtracks, in Figure 2 they are mixed up using a randomly generated matrix A (in this specific case scenario we have $m = 3, n = 3$), and in Figure 3 we can observe a recovery attempt with a high learning rate of 0.01 and 5000 iterations. In this example, A mostly ended up mixing the first original soundtrack with the third, and the second original soundtrack, initially noisy, seems to have been "diluted" in the others.

B. Quantifying the performance

I found 3 metrics to estimate the performance of my algorithm. For the first one, visual, I chose to superpose the best correlated signals U and Y in a single plot after normalizing them in order to compare similar magnitudes. As shown in Figure 4, the original signal is shown in green and the recovered signal in red. I set the two colors as

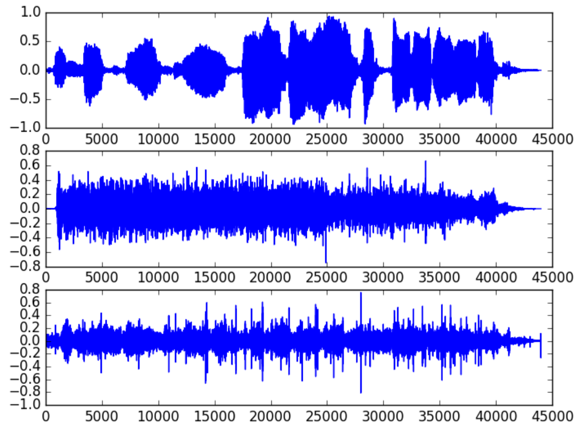

Fig. 1. Original signals

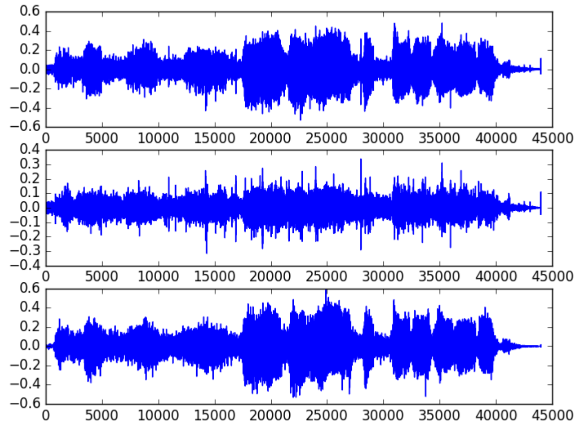

Fig. 2. Mixed signals

slightly transparent so that when the two graphs superpose well, it becomes apparent as the colors fuse into brown. At first glance the results look good for our demonstrated performance, the superposition is well matched and the correlation factors are high. However, seeing the red noise in the first plot, we understand it does not look perfect.

This lead me to the second metrics, the audio itself. As we can verify by listening to the resulting audio files, what seems to be noise in the first recovered soundtrack is actually the strong motor noise from soundtrack 2 which is still partially mixed.

The third metrics which I found the most useful, is the correlation factor r for a given original soundtrack s_U against a predicted recovered s_Y:

$$r = \frac{\sum(s_U(i) - \overline{s_U}))(s_Y(i) - \overline{(s_Y)})}{\sqrt{\sum(s_U(i) - \overline{s_U})^2 \sum(s_Y(i) - \overline{s_Y})^2}} \tag{13}$$

As we can see in Figure 5, I checked the evolution of the

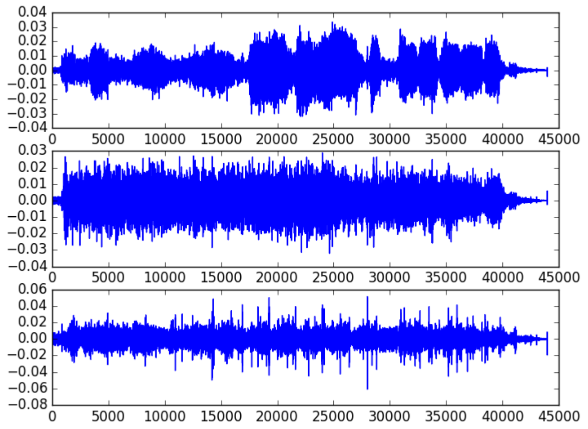

Fig. 3. Recovered signals

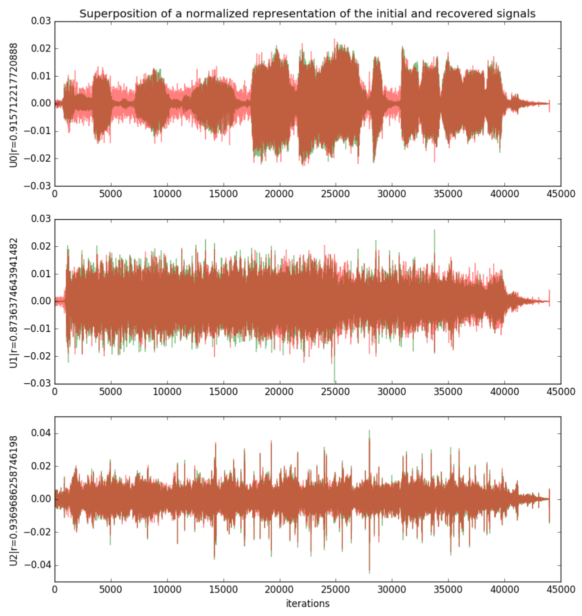

Fig. 4. Superposition of recovered and original signals

correlation factor over the iterations to better understand what are the critical steps toward convergence. I turns out that approximately 1250 iterations reached a limit in the performance.

C. Extended testing

As of how to improve the results, I have increased the number of iterations to 50000 and tried to reduce the learning rate to 0.005. An example of the results is shown in Figure 6, I have considerably improved the correlation factors of each recovered signal and the noise we previously described in the first recovered soundtrack is much more attenuated. Further reducing the learning rate slows down

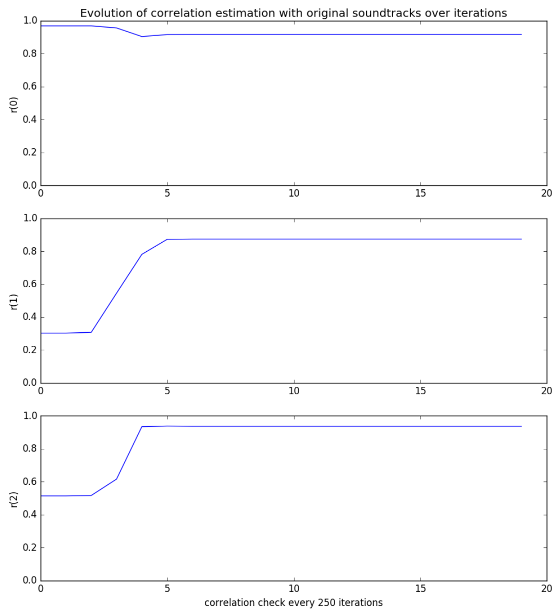

Fig. 5. Correlation estimation

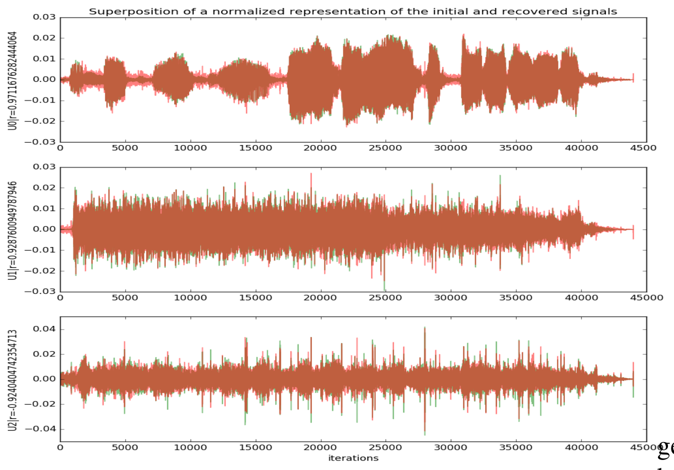

Fig. 6. Better correlations

the progression considerably, and increasing the number of iterations becomes very time-consuming.

IV. Updated results and algorithm

We slightly modify the algorithm with:

$$\Delta W = \alpha(It + (1 - 2Z)Y^T)W \tag{14}$$

This leads to much faster convergence with correlations reaching 0.9999 using a learning rate of 0.0000001. On figures 7 to 10, we notice that he noise has almost completely disappeared and cannot easily be noticed with any of the metrics mentioned above.

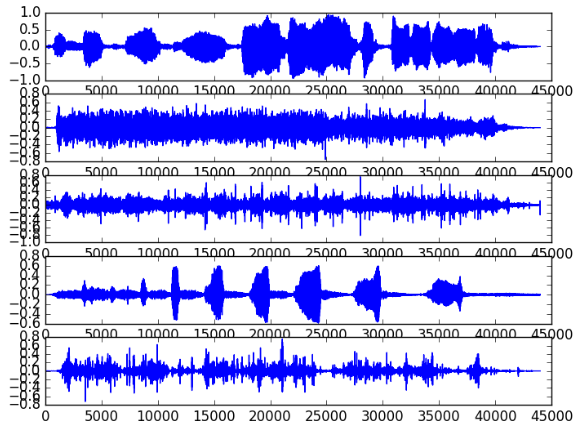

Fig. 7. Original signals

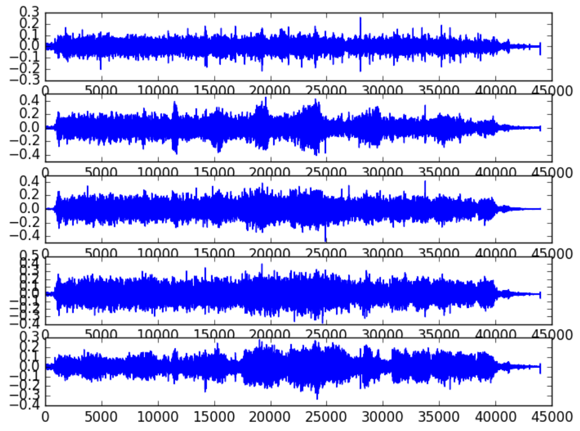

Fig. 8. Mixed signals

V. Conclusion

I have tried improving the performance of the system by generating an adaptive learning rate α, however I haven't been able to significantly improve the algorithm's performance using it. I believe it is much more efficient to try different constant values as it consists in one parameter, rather than the three parameters α_{min}, α_{max} and N mentioned in Equation 12.

Moreover, the algorithm is much more sensitive to the order of magnitude of the learning rate than its subtle variations, too high can make the algorithm diverge severely and too low may completely freeze the correlation's improvements.

However, I still believe it is possible to boost the performance of this gradient descent by adapting the learning rate. An example would be by providing impulses when the correlation factor seems to be stuck, in order to avoid potential local minimum.

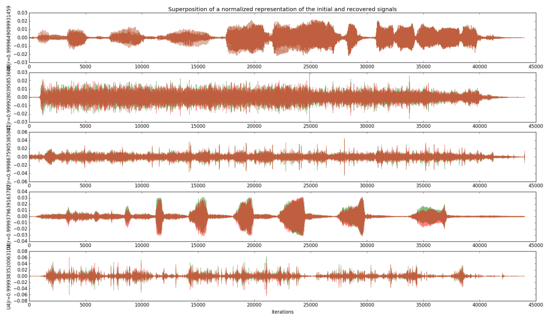

Fig. 9. Superposition

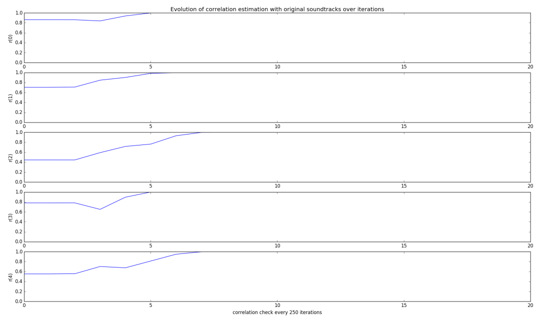

Fig. 10. Correlation evolution