Accomplishing Blind Source Separation using Independent Components Analysis (ICA)

Yue WU

yw9998

sophywu@utexas.edu

CS 391L

2019

February

1 Introduction

This homework aims at using Independent Components Analysis (ICA) to accomplish Blind Source Separation. We will first mix n = 5 original signals into m = 8 mixed signals, and then reconstruct the original signals from the mixed signals by applying ICA.

2 Methods

We first mixed n = 5 original signals stored in matrix U into m = 8 mixed signals stored in matrix X by multiplying the matrix U with matrix $A \in \Re^{m*n}$ (Equation 1). Values in A were chosen randomly and were ranged between 0 and 1. A indicates how much each original signal is contributing to each mixed signal.

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

Then we started to reconstruct the original signals. We first initialized a random matrix $W \in \Re^{n*m}$. Values in W ranges between 0 and 0.1. Multiplying W with X gave us reconstructed signals stored in matrix Y (Equation 2). The matrix Y is of the same size as the matrix U.

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

However, after the very first step the matrix Y was still very different from the matrix U. Therefore we need to apply the algorithm to keep updating the matrix W until the matrix Y becomes similar to the matrix U. The algorithm consists of the following two equations. Equation 3is used to calculate gradients of every element in the matrix Y and Equation 4is used to calculate the amount of update, where η represents the learning rate and t represents the number of values in each signal. We divided t here to prevent overflow. We will discuss how the choosing of η can affect our results in the following section. We then update the matrix W using Equation 5. After that, we checked the convergence condition by calculating the differences between the current recovered signal and the previous recovered signal (Equation 6). We found out the maximum value in the matrix C and compared it with a "judging" value we set at the beginning of the iterations. When the maximum value in the matrix C is smaller than the "judging value", we consider the iteration has converged and completed. The last step is noting down the number of iterations we took to arrive at the "judging value".

$$z_{i,j} = g(y_{i,j}) = \frac{1}{1 + e^{-y_{i,j}}} \tag{3}$$

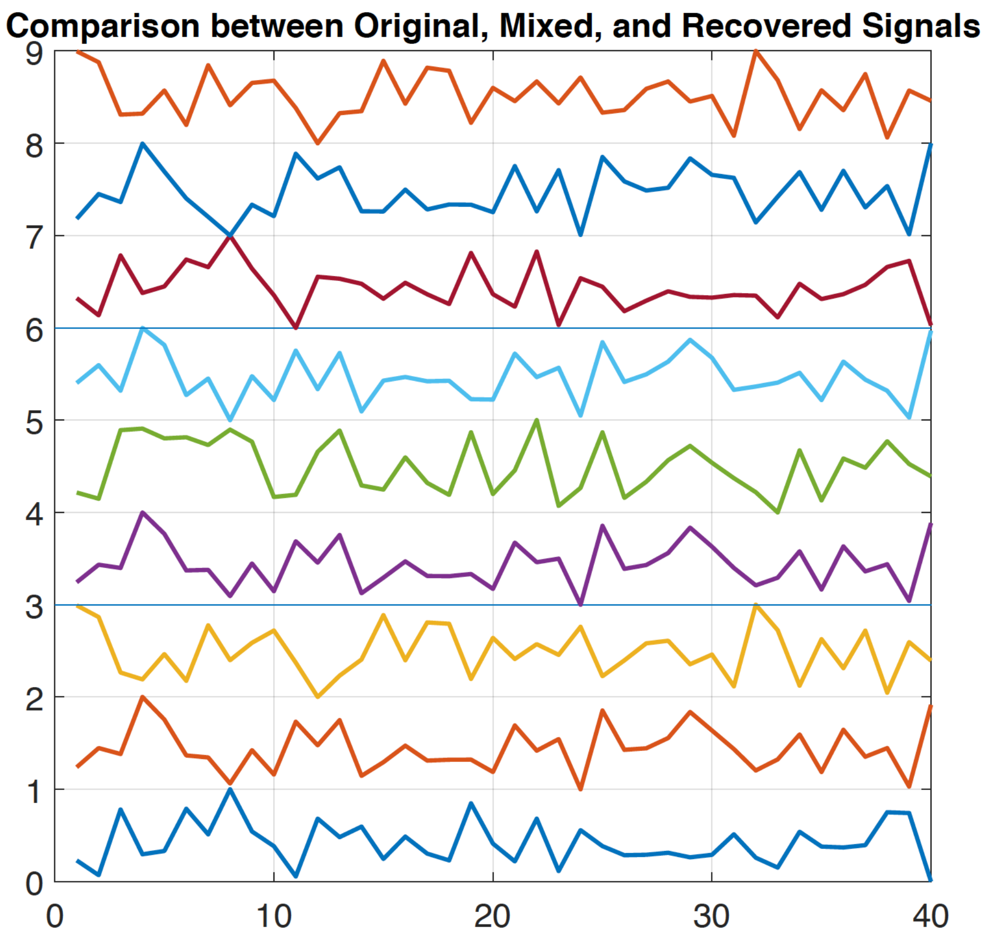
$$\Delta W = \eta(I + \frac{(1 - 2Z)Y^T}{t})W \tag{4}$$

$$W = W + \Delta W \tag{5}$$

$$C = abs(\Delta W * X) \tag{6}$$

3 Results

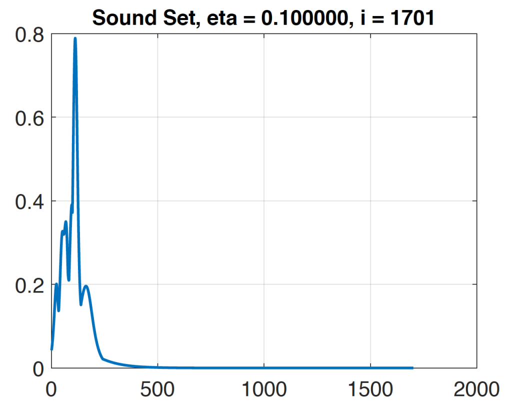
We first tested our algorithm using the test set (Figure 1). The top three lines are the recovered signals; the three lines in the middle are the mixed signals; the three lines shown at the bottom are the original signals. By setting the learning rate η to be 0.01, we nearly fully recovered the signals although the sequence was different from the sequence at the beginning.



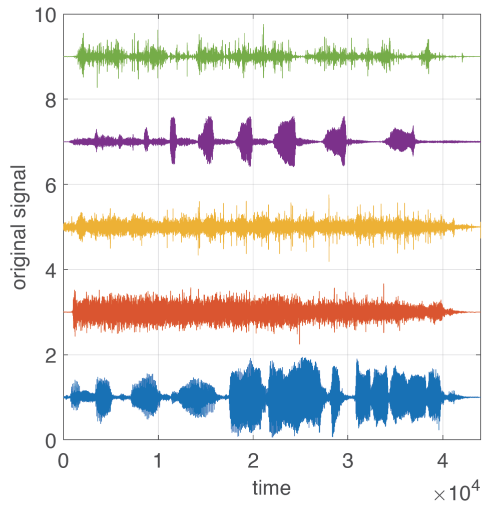
(a)

Figure 1: Results from the test set. The top three lines are the recovered signals; the three lines in the middle are the mixed signals; the three lines shown at the bottom are the original signals.

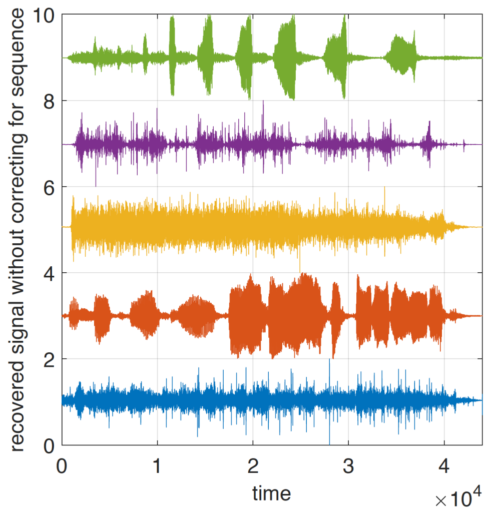
We then mixed all five original signals into eight mixed signals and reconstructed the original signals using the algorithm. We tested for three ηs: 0.1, 0.01, and 0.001. We recorded the variation of the maximum value in matrix C with iterations. When the maximum value started to change slowly and finally reach the "judging value", the iterations stopped, and we considered the process completed. Since the restored signals might be in a different sequence as the original signals, we compare the shape of each restored signals with the shape of each original signals by calculating the correlation coefficients. Since the "judging values" we set are typically very small, when the restored signals matched with the original signals, the correlation coefficients can reach 0.9999. After matching the signals, we normalized the restored signals so that their values still ranged between -1 and 1. The results are shown in Figure 2, Figure 3 and Figure 4 for three distinct ηs.



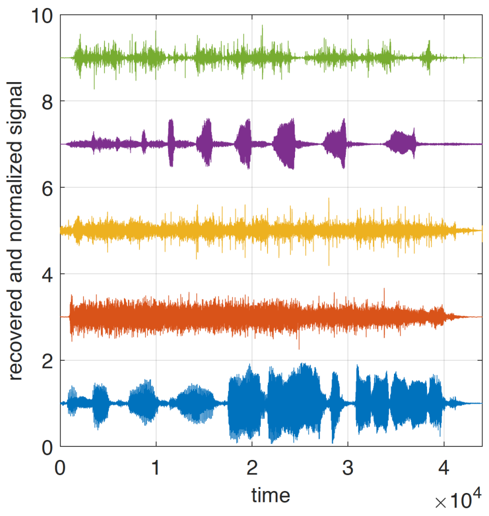
(a)



(b)

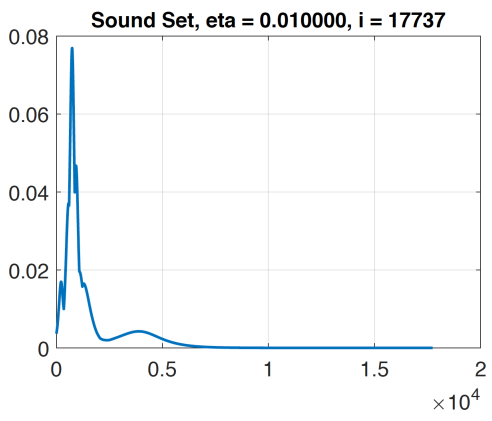


(c)

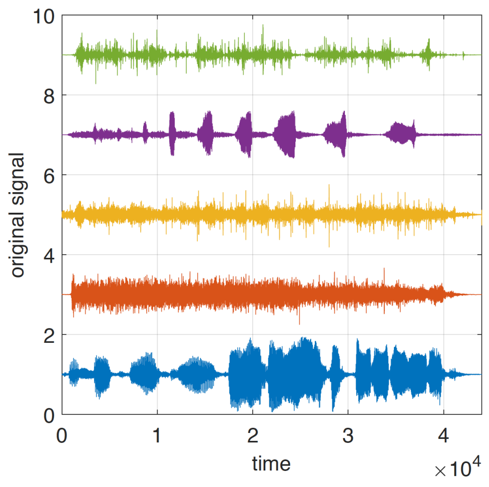


(d)

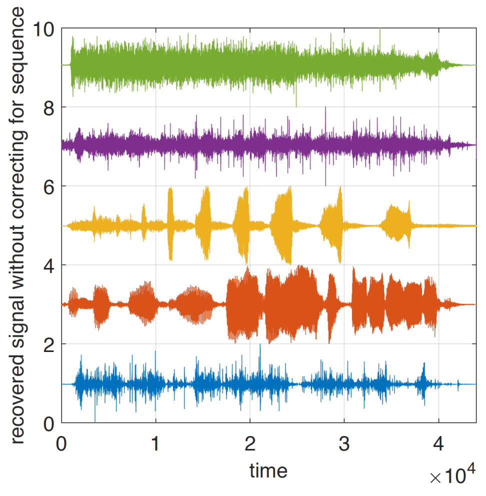
Figure 2: This figure shows the situation where the learning rate is 0.1. (a) Variance of the difference between the current recovered signal and the previous recovered signal with iterations (b) Original signals (c) Restored signals before correcting for sequence and normalization (d) Restored images



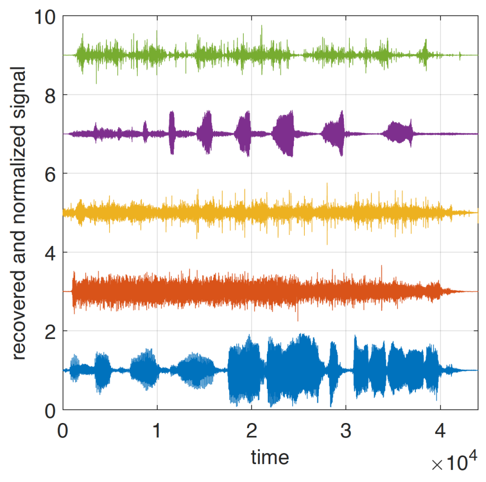
(a)



(b)



(c)



(d)

Figure 3: This figure shows the situation where the learning rate is 0.01. (a) Variance of the difference between the current recovered signal and the previous recovered signal with iterations (b) Original signals (c) Restored signals before correcting for sequence and normalization (d) Restored images

4 Conclusion

We have mentioned in the previous section that when the restored signals matched the original signals, the correlation coefficients can reach 0.9999. This indicates that the restored signals were very similar or almost the same as the original signals. This remains true for

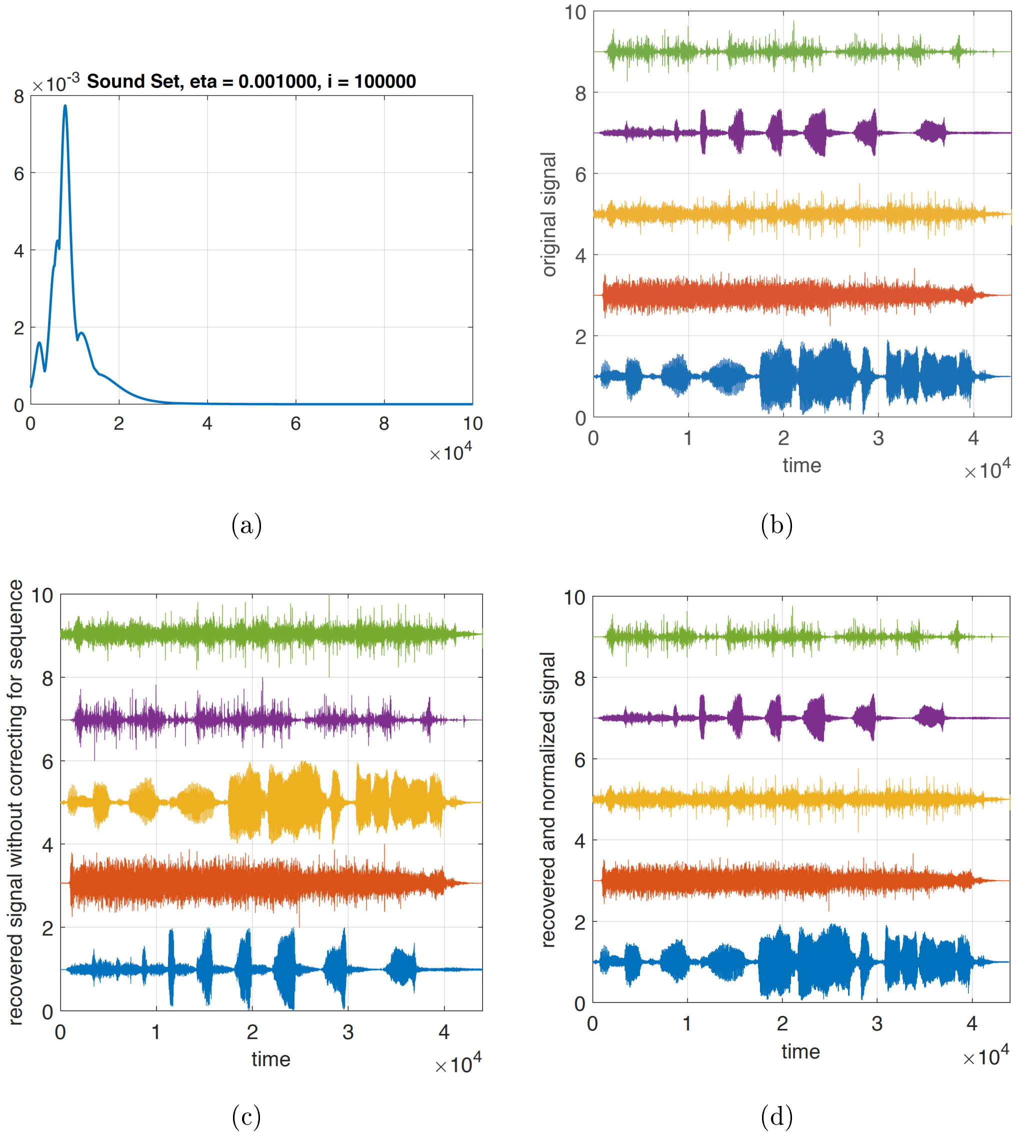


Figure 4: This figure shows the situation where the learning rate is 0.001. (a) Variance of the difference between the current recovered signal and the previous recovered signal with iterations (b) Original signals (c) Restored signals before correcting for sequence and normalization (d) Restored images

all three ηs.

Convergence of Gradient Descent varied with regard to learning rate η as is shown in Figure (a) in Figure 2, Figure 3 and Figure 4. We set the "judging value" to be 10^{-9}, which means that when the maximum value in the matrix C is lower than this value, or in other

η	Maximum value in the matrix C	# of iterations
0.1	10^{-9}	1701
0.01	10^{-9}	17737
0.001	10^{-8}	100000

Table 1: The optimal number of eigenvalues used for different numbers of training samples as well as the corresponding accuracy.

words, when the difference between the current recovered signal and the previous recovered signal is smaller than this value, the iterations would stop, and we considered that we had reached convergence. When $\eta = 0.1$, we reached convergence after only 1701 iterations. When $\eta = 0.01$, we reached convergence after 17737 iterations. When $\eta = 0.01$, we did not reach the "judging value" after 100000 iterations. The final difference that we arrived at after 100000 iterations was only 10^{-8}. To conclude, as the learning rate η decreases, it took longer to reach convergence (Table 1).