CS 391L HW2 Independent Component Analysis

Haocheng An(UTEID: HA8886)

I Introduction

Independent Component Analysis is a great tool in machine learning. The key point behind it lies on the key word Independent. That is, the materials we got is actually a mixture of some independent component. In order to recover the original component, we need to do such analysis. This analysis has a wide range of usage. A good use would be to recover the sound we have to its source. In this lab, I tried to mimic this process. Faced with music which represented as matrix, where each row represents a source of the sound. We implement this technique for this lab and conducting some experiments on it. We compare the input and output graph and see how well the recovery goes. Also, we test on the efficiency. By using different learning rate, we try to find how many iterations does it take to converge.

II Methodology

Denote the input set as U.

We first mix the data set: X = AU. A is a random matrix that takes number ranging from [0,1]. This process imitates the mixing result and thus we get X as the result of mixing.

Initialize W with small random values with uniform distribution with ranging from 0 to 0.1 as our current approximation of A^{-1}.

Run following algorithms for multiple times.

Later create delta W that add to W so that we can update our approximation every time. So our initial guess is Y = WX

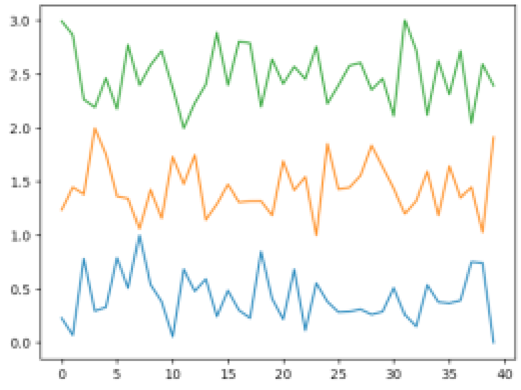
Now create Z with logistic regression. Calculate $z_{i,j} = 1 + e^{-y_{i,j}}$

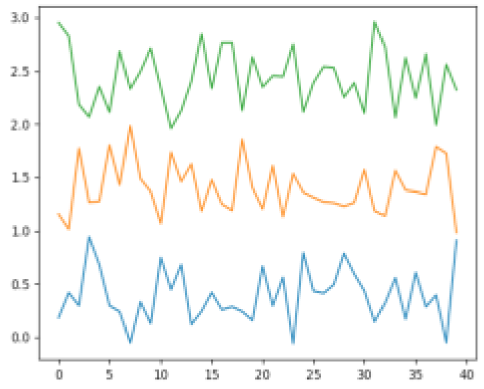
Calculate delta W = yita*(I + (1- 2*Z)*Y')*W. yita is the small learning rate that we pick. With different learning rate, we may need different number of iterations to converge. Initially we set as 10**-5, then we change different values to 10**-3 with equal distance between two points and gap set as 5*10**-5, we get different number of steps to converge to a specific value. Once converge, we normalize the graph back. That is, set the min value to -0.5 and max value to 0.5 and scale every point linearly. In this way, we recover the graph we need.

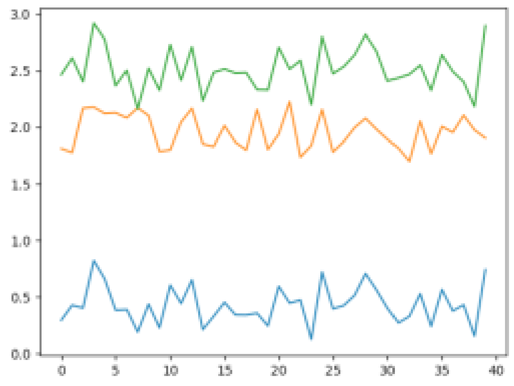
III Result

1) The waves pattern for the small test set and big test set.

For the smaller test set, I set the exact parameters as the handout did—set 0.01 as the learning rate and 1000000 as number of iterations. The graph is shown as below:



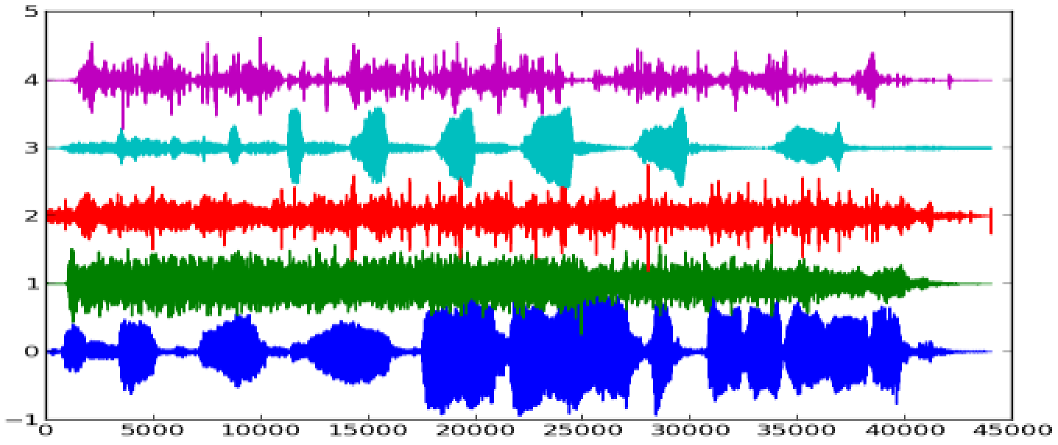


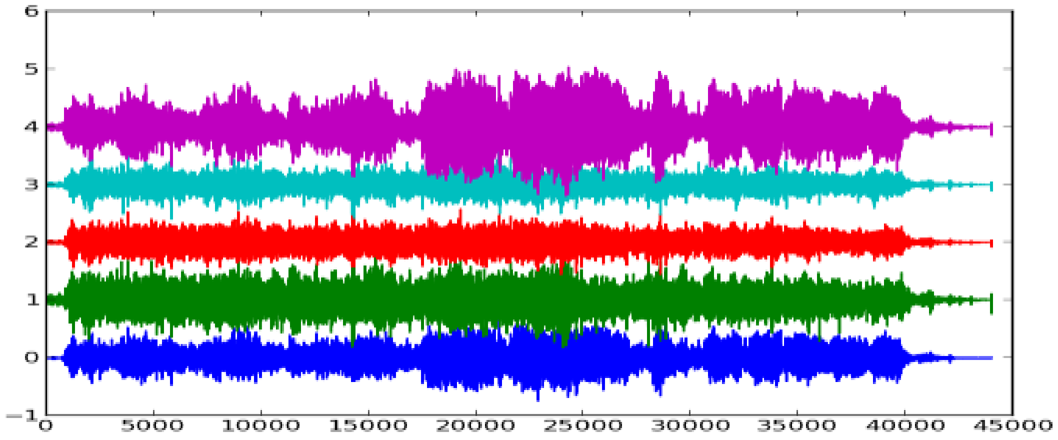


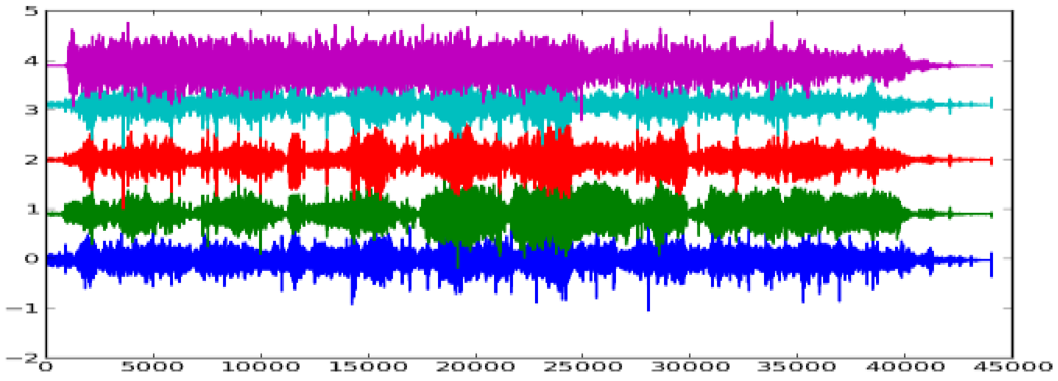
The top left graph is the original data, the bottom left is the mixed data and the top right is the recovered data.

We can see the recovery generally works fine. However, the order they appear may change. As plot shown above, the yellow line at the left corresponds to the blue line at the right while the blue line at the left corresponds to the yellow line at the right.

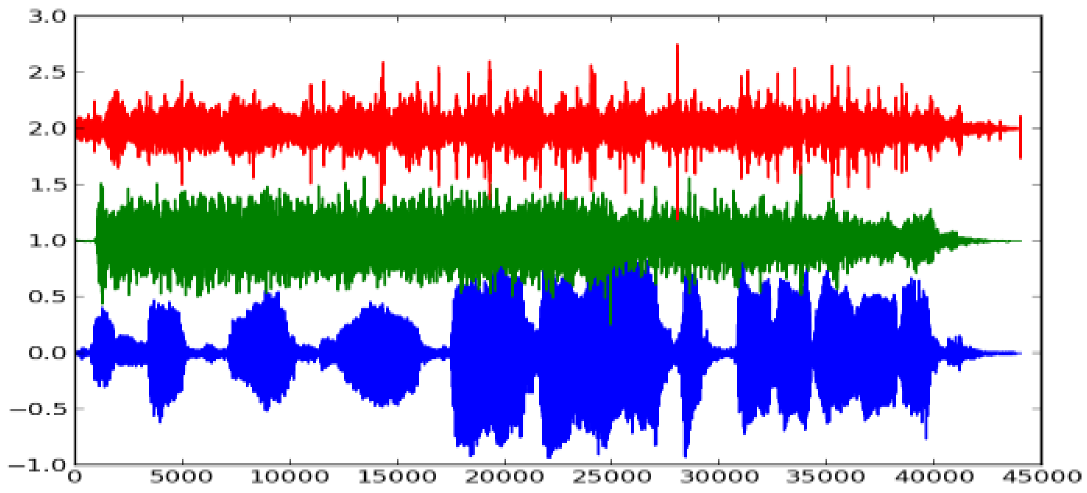
Let us analyze on the big data set as a big picture first:

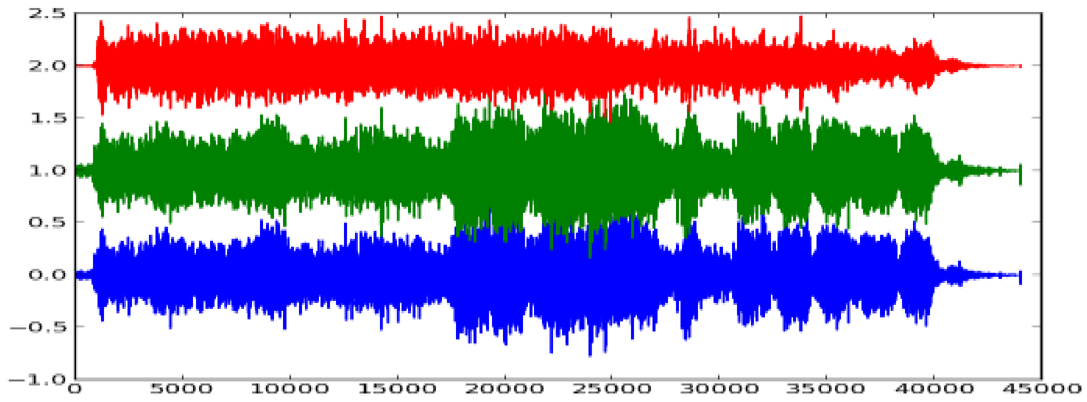


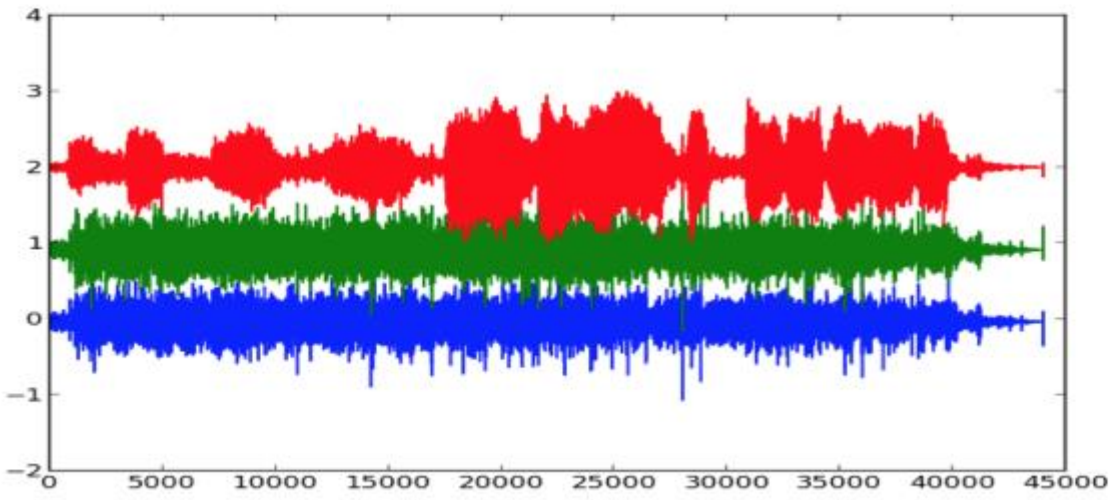




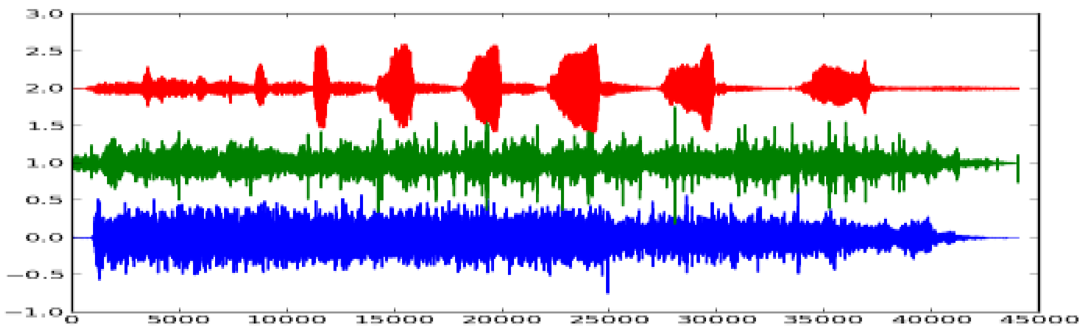
The graphs above from top down is reflecting initial data, mixed result and the recovery result and all graphs below follow this pattern. As we can see, the initial data red line that was plotted with center at around 2 was transformed to the recovery data at center about 0. The one around 1 was transformed to 4. The other sound record, however, are not easy to distinguish the recovery ones. This means we experience some loss on mixture part.
Also, different combinations of this sounds give us different result. We extract 3 vectors from the original matrix and see their effects:
The combination of 0^{th}, 1^{st} and 2^{nd} vector before(graphs on the next page):
We can recognize that the red record at the bottom is the blue record at the top and the blue record at the top is the red record at the bottom. In this way, we think the program performs well on this recovery.

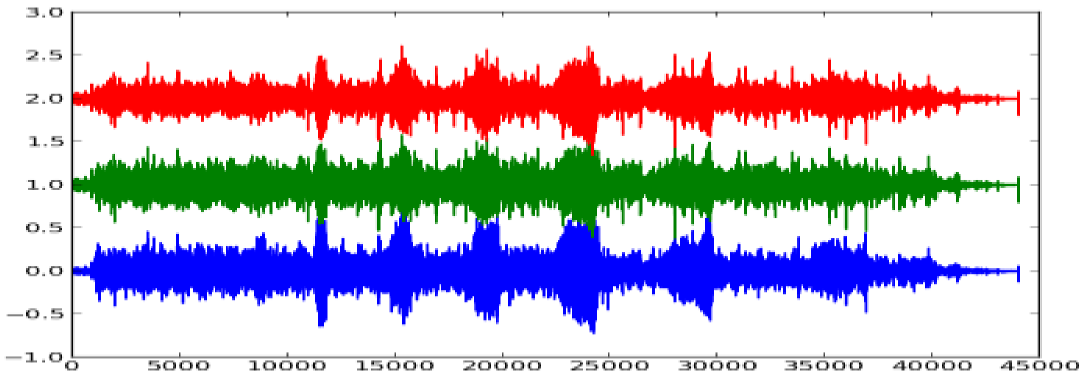

3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
5000
10000
15000
20000
25000
30000
35000
40000
45000

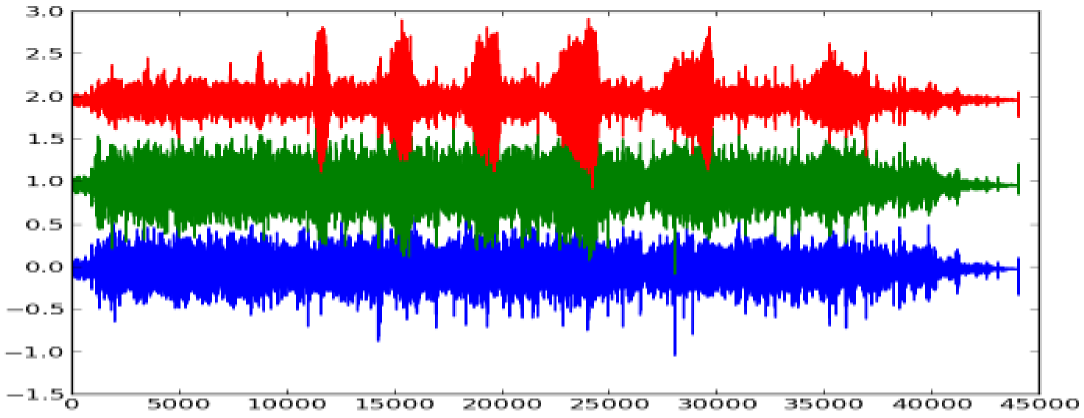

2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
5000
10000
15000
20000
25000
30000
35000
40000
45000


4
3
2
1
0
−1
−2
0
5000
10000
15000
20000
25000
30000
35000
40000
45000

Let us discuss about vector coming from 1st , 2nd and 3rd row:
It has similar result as the one above. This time, the red records keeps stable but the green and blue records switch their order.



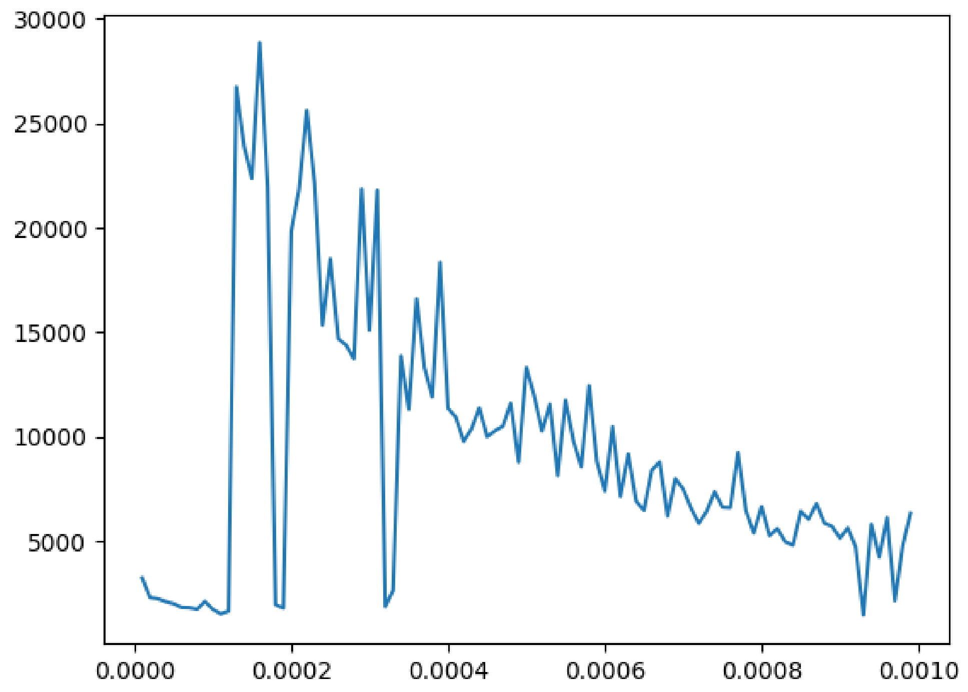




So for different combinations of the sound we pick, they usually leads to different result. In general, the less the value of the source file, the less the length of the input file, the easier we can recover the original video and get better result.

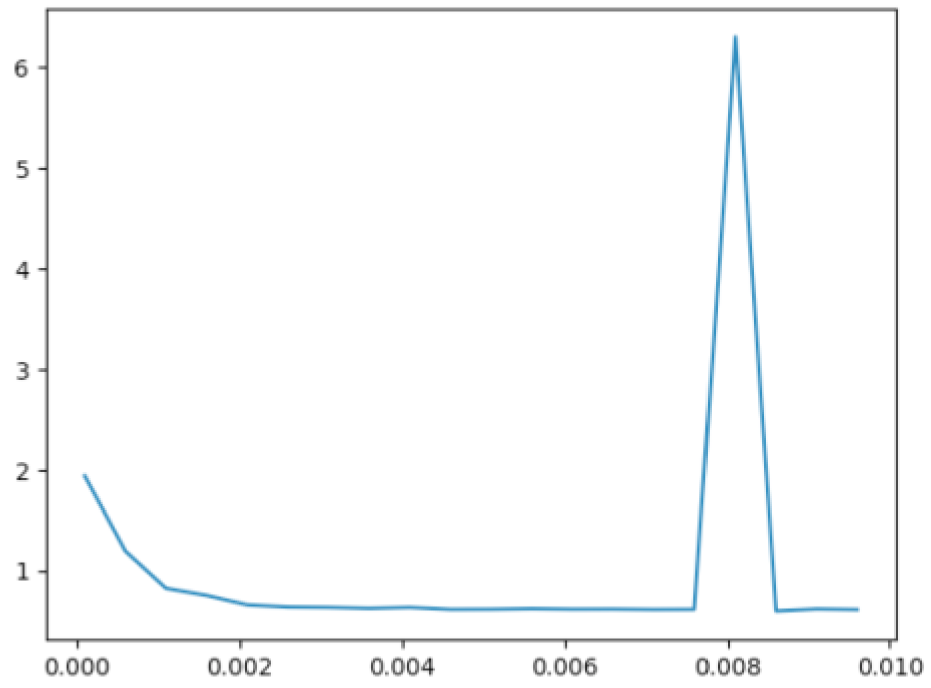
2) For the following part: we only care about the smaller testing set as the bigger set is too slow in running:

   a)For different learning rate, usually we need different iterations for convergent. Here, we use the fubini's norm, which is the square sum of every element in the matrix W*X, representing the recovering sound. Every time, we update W by adding the gradient delta W to it. We keep track of the norm difference in two adjacent iterations and stops once the difference is smaller than 10^-4. The number of iterations needed can be seen as below:



   In general, with the increasing sequence of the learning rate, the number of iterations needed is decay. However, it is not necessary that bigger the learning rate, the smaller number iterations needed. It actually jumps back and forth many times. This occurs maybe because we cannot reach the extrema point directly, rather, just jumping around the extrema point with little steps forward to the target.

b) The graph below shows the absolute error corresponding to learning rate starting from 10**-4 to 10 ** -2 with equal distance 5*10**-4. As the recovered music may have different order from the input order, we consider all 6(3!) combinations of our recovery. Comparing the absolute difference of each vector recovered with original vector and get their l2 norm, we sum all square root of square of three l2 norms and get the graph below.



We can see that generally the absolute error is around 0.5. Given the l2 norm of a vector is around 12~13. This gives us about 4% relative error. Also, the spike occurs when learning rate is about 0.008 is worth noticing. The bigger learning rate sometimes means you have a relatively large error.

IV Summary

In all, through the process of this lab, we have found that independent component analysis works well for smaller testing set and some of the big set.

For smaller data set, we can see that different learning rate will have different effect on the speed of convergence and the error it brings to the recovery. In general, a relatively high learning rate would cause the error relatively small and we need smaller number of times to converge. However, this only is a trend and some specific learning rate would generate better result than others.

V Reference

Scipy wav file function
https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.io.wavfile.read.html