Machine Learning Assignment 2

Huihuang Zheng, huihuang@utexas.edu

hz4674 Spring 2016

1 Using Code

To run my code, open **src/main.m**, change the **DATA_DIR** to where you put your **icatest.mat** and **sounds.mat**, then run the *src/main.m* with Matlab. It will output one figure of mixed signals and one figure which put origin signals and estimated signals together. For comparison of origin signals and estimated signals, those signals are listed in n rows and 2 columns, where n is the number of signals. The order may not be correct. But you can find the matched shape.

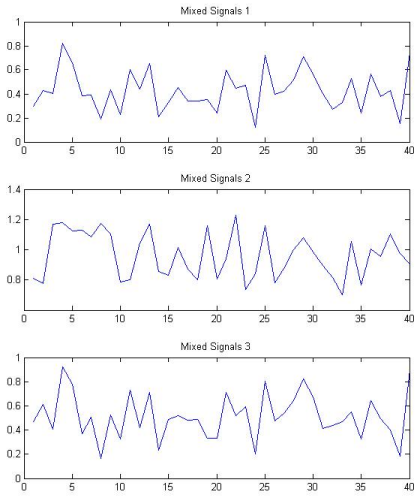
2 Algorithm Implementation

The **src/mixSignals.m** mixes source signals for inputs of source signals and mixing matrix. If you don't specify a mixing matrix, I generate a random one. The **src/ICA.m**) is the function calculating recovered signals and unmixing matrix. Other functions src/rescaleVector.m src/showResult.m and src/showSignal.m are just used for printing results. My algorithm is exactly same as on Canvas assignment page. But I change something about the learning rate. In website page, it suggested that when we deal with the large data set **sounds.mat**, our program may be slow, then we can set smaller learning step. Let our unmixing matrix in i-th iteration be W_i. Let the norm of difference of unmixing matrices in two neighbor iteration, which is $norm(W_i - W_{i-1})$ be Δ. Let the converge threshold be H, then the algorithm stops when $\Delta < H$. In my experiment, I found when we set smaller learning rate, it still converge slow (about ten thousands of iterations if I set $H = 1e^{-9}$). I did this in different way: I set a large learning rate, but after a number of iterations, I decrease the learning rate, which makes converge faster than smaller learning rate at beginning and can converge with smaller learning rate at the end. I set initial learning rate to 0.1 and for every 1000 iterations, I multiply learning rate by 0.1. Be doing so, my program just converge at thousands of iterations.

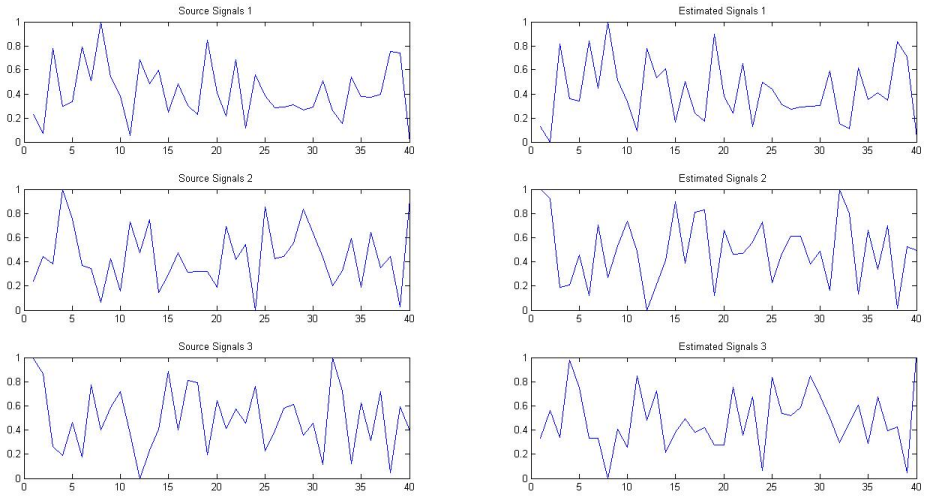
3 Experiment

3.1 Small Dataset

I run experiment using **icatest.mat**. My program runs pretty well in this dataset. See result:



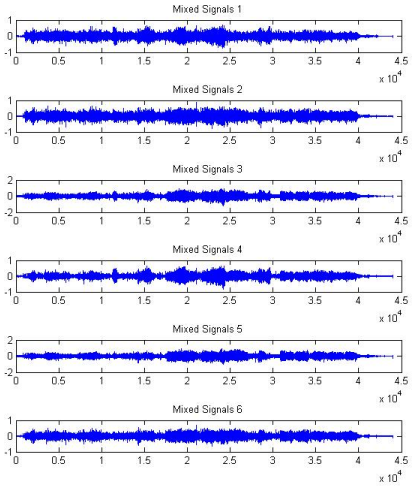
Mixing Signals



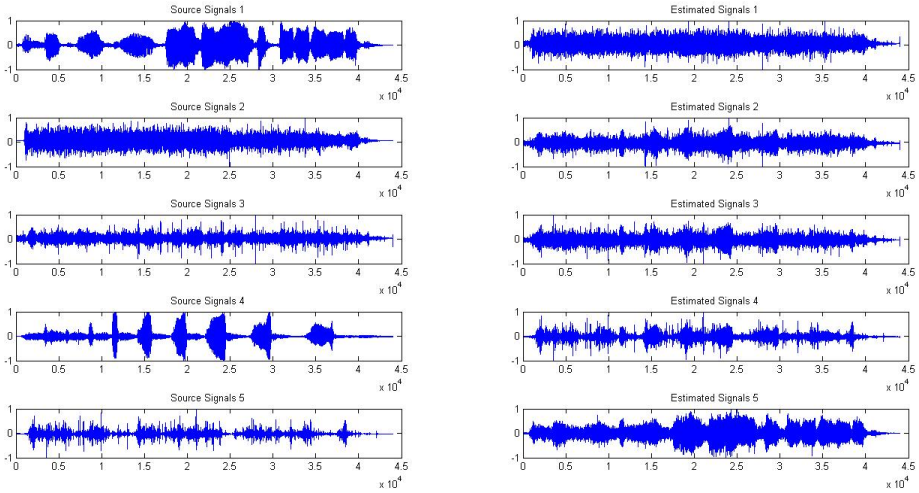
Origin and Recovered Signals. The Estimated Signals 2 and 3 are corresponding to Origin Signals 3 and 2.

3.2 Large Dataset

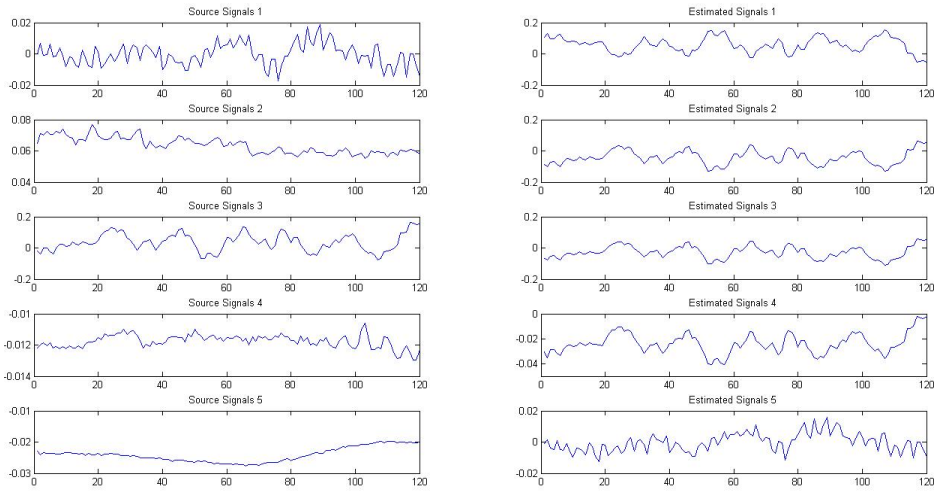
For the large dataset **sounds.mat**. My program cannot run very well. But I also do experiments about why the ICA doesn't work well. First, see my result for experiment of **sounds.mat**:



Mixing Signals of 5 Signals sounds.mat to 6 received signals



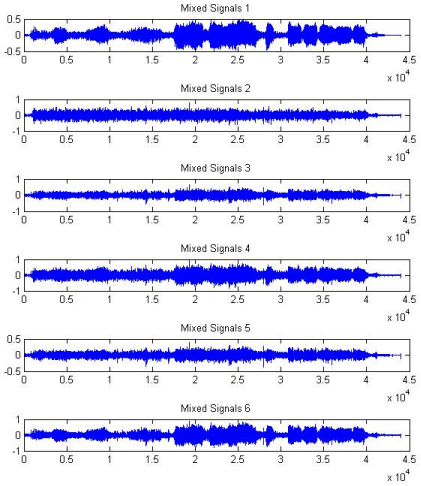
Origin and Recovered Signals.



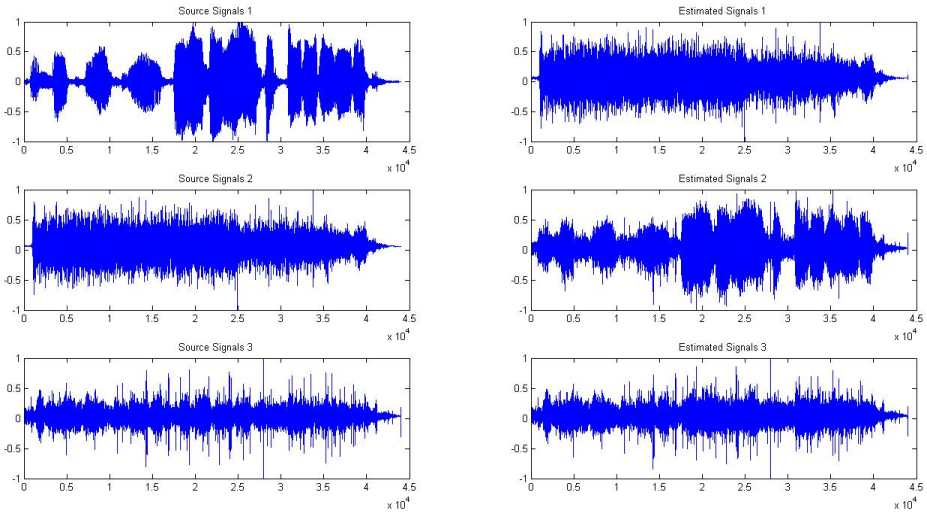
Snapshot of origin and recovered signals when $t <= 120$

The matrix sounds is $n * t$ matirx (number of signals multiplies time). Then I would think, is the ICA cannot work well for larger n or larger t. I did experiment about just use first n signals or first t signals. My experiment result suggests that ICA can work well with larger t, smaller n. But it cannot work well with larger n:

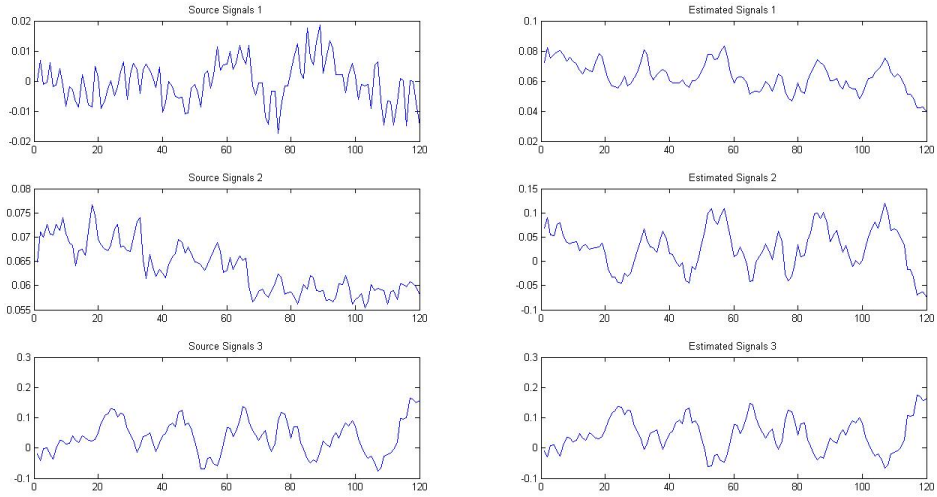
3.2.1 $n = 3, t = 44000$



Mixing Signals of First 3 Signals sounds.mat

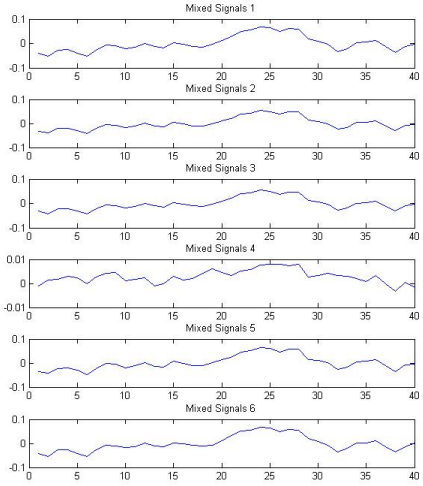


Origin and Recovered Signals. The Estimated Signals 1 and 2 are corresponding to Origin Signals 2 and 1.

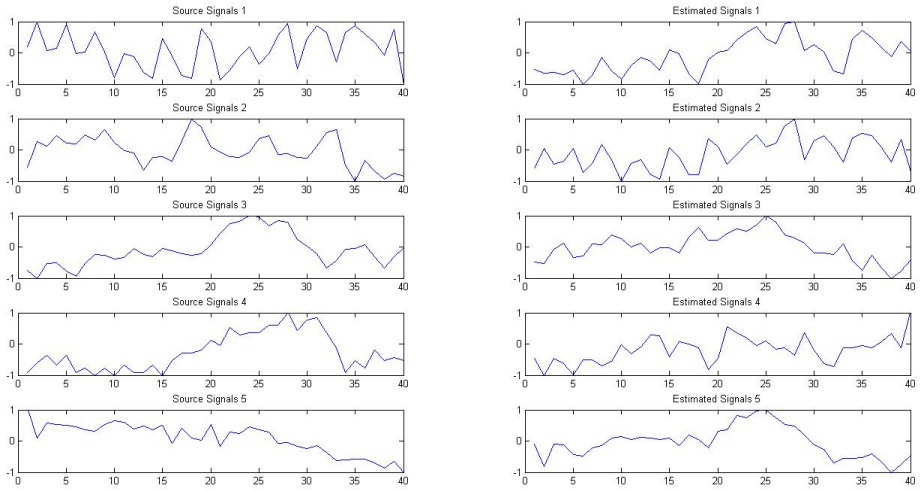


Snapshot of origin and recovered signals when $t <= 120$

3.2.2 $t = 40, n = 6$



Mixing Signals of First 3 Signals sounds.mat



Origin and Recovered Signals.

From the experiment we can see ICA can work well with larger t, smaller n. But it cannot work well with larger n even with small t.

4 Conclusion

In this assignment, I not only implemented the ICA algorithm, but 1, find another way to speed up converge in ICA. 2, find ICA cannot work well for larger number of signals, but can work for larger time.