Applied Information Theory 691GG

http://www.cs.umass.edu/∼elm/Teaching/691GG_F10/

Assignment 3: Independent Components Analysis and Blind Source Separation

In this assignment, you will be using independent components analysis to solve the blind source separation problem. You will be given 3 short recordings: mix1, mix2, and mix3. Each one lasts just a second or two. Each recording is a different linear combination of the same 3 original recordings: A, B, and C. Your job is to unmix the recordings to produce the original recordings.

I recommend matlab for this assignment, but you can do it in C++ or Java if you want. If you use C++ or Java, you'll be on your own about to play audio files. I give instructions on how to play audio files in Matlab.

This should go without saying, but **YOU ARE NOT ALLOWED TO USE ANY CODE FROM THE WEB FOR THIS ASSIGNMENT. YOU ARE ALSO NOT ALLOWED TO LOOK AT ANY CODE FROM THE WEB.** In other words, you need to do this from what you learned in class, and from the paper posted on the web site.

Entropy estimators

This assignment will require you to write a function that estimates the entropy of a *one-dimensional* distribution given a sample from that distribution. While theoretically you could use the plug-in estimate described in class, but it will be very slow. Instead, you should use the technique from my paper on ICA, which is on the course web page. This is called the *m*-spacing estimate of entropy. Once you understand it, it is very easy to implement, requiring only a few lines of matlab code.

Now I describe the steps you should follow to write a working blind source separation program.

1. Start by downloading the 3 mixed audio files from the course web site. If you are using matlab, just load each file by issuing the commands
   ```
   load mix1
   load mix2
   load mix3
   ```
   This will create arrays of samples with the names mix1, mix2, and mix3. If you are using C++ or Java, you will want to use the versions of the files labeled mix1ascii, mix2ascii, and mix3ascii. These files just give each number in a space separated file, so you'll need to write code to read these into an array in Java or C++.

2. Try playing them in matlab with the commands
   ```
   soundsc(mix1, 44100);
   soundsc(mix2, 44100);
   soundsc(mix3, 44100);
   ```
   The number 44100 is the sampling rate of the audio files. You should hear various mixtures of voices, but you may not be able to sort out exactly what is being said.

3. The good news for you is that I have already decorrelated the audio signals. As I discussed in class, this means that you can limit your search over unmixing matrices to rotations. Since there are 3 Jacobi rotations in three dimensions, you will be applying these rotations to the mixture signals and trying to find rotations which minimize the mutual information among the resulting signals. The 3 rotations you will be using include rotation in the $x-y$ plane:
   $$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

rotation in the $y-z$ plane:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{2}$$

and rotation in the $x-z$ plane:

$$\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}. \tag{3}$$

Write function in matlab to return each matrix, given a particular amount of rotation θ.

4. You will be using the rotation matrices to transform the data. To do this, you will need to make the points from the 3 audio files into a matrix with 3 rows and 100000 columns. You do this simply by writing:
   `data=[mix1; mix2; mix3];`
   You will implement the following algorithm. For each rotation matrix, for each angle (you can step by about 1 degree in the angles), apply the rotation matrix to the data and look at the rows of the resulting matrix. You can extract the row of a matrix like this:
   `rowi=data(i,:);`
   Evaluate the sum of the entropies of these rows. Over all rotations of a particular matrix, choose the one which makes the sum of the entropies lowest. Call this best rotation matrix R_i, where i is the index of the step you are on.

5. Now that you've found the best angle θ for a given rotation matrix, you want to process the data to produce
   `dataNew = Ri * data;`
   and then repeat the process again with the next rotation matrix.

6. I suggest that you repeat this optimization *twice for each matrix type.* In other words, do the following sequence of optimizations:
   $R_1 = Rot_{xy}$
   $R_2 = Rot_{yz}$
   $R_3 = Rot_{xz}$
   $R_4 = Rot_{xy}$
   $R_5 = Rot_{yz}$
   $R_6 = Rot_{xz}$,
   each time creating a new data matrix. When you are done, the final matrix should have the unmixed signals.

7. Try playing the unmixed signals. If all worked well, you should hear two recordings which feature only my voice, and one recording which features my daughter counting to 2.

8. Finally, and this is the most important part, you need to give me the rotation matrix that produced your final result. In matlab you can do this by computing
   `Rfinal=R6*R5*R4*R3*R2*R1;`

You should turn in 2 things. The first is your matlab code (or c++ or Java). The second is your final rotation matrix.