HW4 Independent Component Analysis

Due: Tuesday, February 21, 2012 at 11:59pm

Independent Component Analysis (ICA) is an algorithm for accomplishing Blind Source Separation. That is, if there exists an **n** by **t** matrix **U** of **n** source signals of length **t** (in this case, assumed to be functions of time, such as sound, although that is not a requirement), and you have an **m** by **t** matrix **X** of **m** mixed signals (**m>=n**) of length **t** that consist of different linear mixtures of **U** (i.e., **X = AU** where **A** is an **m** by **n** matrix such that $A_{i,j}$ is the weight of the j^{th} source signal in the i^{th} mixed signal), then, under certain conditions, you can recover the original signals **U**, up to a scale factor.

This is accomplished by assuming that there is no correlation between source signals and so any correlation between different mixed signals is due to a common signal showing through the mixture. Our task is to find a matrix **W** that recovers the original **n** source signals (possibly in a different order and with different scale factors).

There are several algorithms for decreasing the mutual information between signals, for this project, we will use a gradient descent method as described in class (this algorithm might still need some refinement):

1. Assume **X = AU**.
2. Initialize the (**n** by **m**) matrix **W** with small random values.
3. Calculate **Y = WX**.
   Y is our current estimate of the source signals.
4. Calculate **Z** where $\mathbf{z_{i,j} = g(y_{i,j}) = 1/(1+e^{-y_{i,j}})}$ for **i** ∈ **[1..n]** and **j** ∈ **[1..t]** (where **t** is the length of the signals).
   This helps us traverse the gradient of maximum information separation.
5. Find $\mathbf{\Delta W = \eta(I + (1-2Z)Y^T)W}$ where **η** is a small learning rate.
6. Update **W = W + ΔW** and repeat from step 3 until convergence or R_max iterations (you get bored and decide it is done).

Your assignment is to do the following:

1. **Get data.** To access the signals we have prepared, click on this link. This .mat file will provide you with a variable called sounds that is a 5 by 44000 matrix. Each row represents a ~four second sound clip (sampled at 11025). These signals are not mixed. You will have to mix them yourself. You might not want to mix all five. Maybe you will. Some of them might work better together than others. That is something you can discuss in your report. If you write it back to file, you will want to be sure and scale it back so that it fits between -1 and 1.

2. **Mix the data.** Create a matrix **A** to mix the signals. Output the signals so that you can listen to the result.
3. **Implement the algorithm.**
4. **Test.** See how well you can recover the original signals over several trials. Plot the recovered signals next to the original signals. You might have to trim out a small section of the signals to make a legible plot. Also, see if some signals separate better than others.
5. **Write and submit.** Write, review and submit a brief report detailing what you did, how well it worked, and something neat that you learned, along with your code. Write your name, email address, and EID in the report. Submit using `turnin` to `dkit`. This is **hw4**.

To debug your algorithm on a small set, you can try this one. Click on this link to download "icaTest.mat". Load the file in matlab with `load 'icaTest.mat'`. This will give you two matrices: **U** (3 by 40) and **A** (3 by 3). You can use **A** to mix **U** and get **X** and then work from there.

This is what I got after running it. The bottom signals are the original signals, the middle signals are the mixed, and the top signals are the recovered signals (all signals have been scaled to fit between 0 and 1). Notice that in the recovered signals, the top and bottom signals are reversed from the original signals. Also notice that the recovery is not perfect, but is surprisingly close. I used **η=0.01** and 1000000 iterations.