ICA Image Recovery Demo for P9120

Shuang Wu, sw3140

11/2/2017

1. Introduction

This document shows that how one can use `EBImage` and `fastICA` packages to implement a demonstrative example using independent component analysis (ICA) to extract the souce images from "image mixtures", which are linearly overlayed images. This is a part of the Group E presentation for the P9120 class (Bowen Chen, Runjia Li, Yanjun Liu, Denise Shieh, Karissa Whiting and Shuang Wu). All the R code and the original images can be found on Karissa's github.

2. Setting up the environment

```
source("https://bioconductor.org/biocLite.R")
biocLite("EBImage")
library(EBImage)
install.packages("fastICA")
library(fastICA)
```

Note that the `EBImage` is a package from Bioconductor.

3. Import images and examine them.

```
### Import 4 images
test1=readImage("./Test1_g.jpg")
test2=readImage("./Test2_g.jpg")
test3=readImage("./Test3_g.jpg")
test4=readImage("./Test4_g.jpg")

#display(test1); display(test2); display(test3); display(test4)
plot(combine(test1, test2, test3, test4), all=T) # Combine the images and plot them.
```

Here the `display()` function in `EBImage` will open your image in your default web browser.

4. Make two linear combinations of test1 and test2.

```
### A linear combination of test1 and test2
img1=0.4*test1+0.6*test2
img2=0.7*test1+0.2*test2
plot(combine(img1, img2), all=T)
```

```
## Convert the two transformations to vectors
img1_vec=as.vector(img1)
img2_vec=as.vector(img2)
```

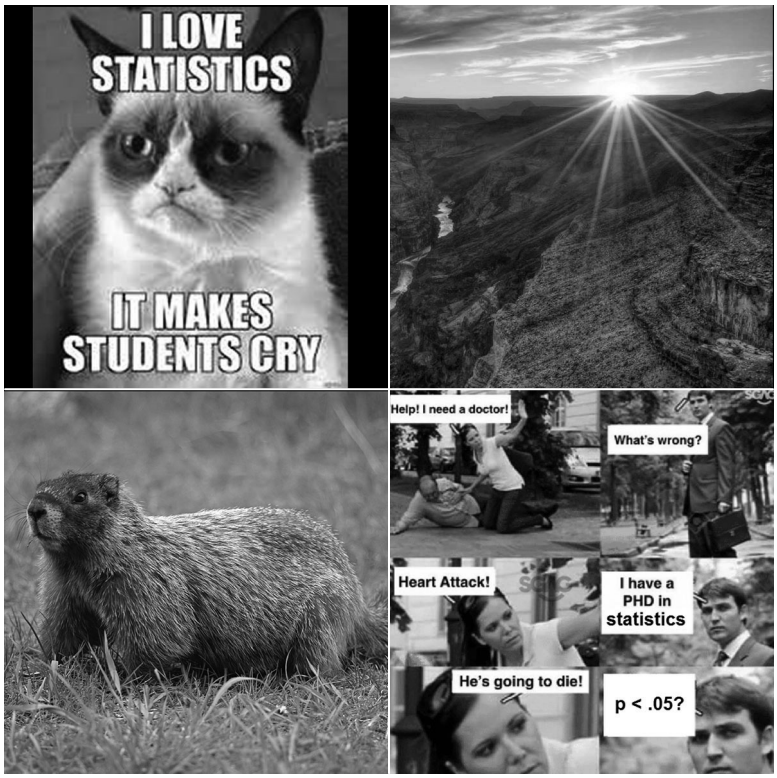

Figure 1: Source images.

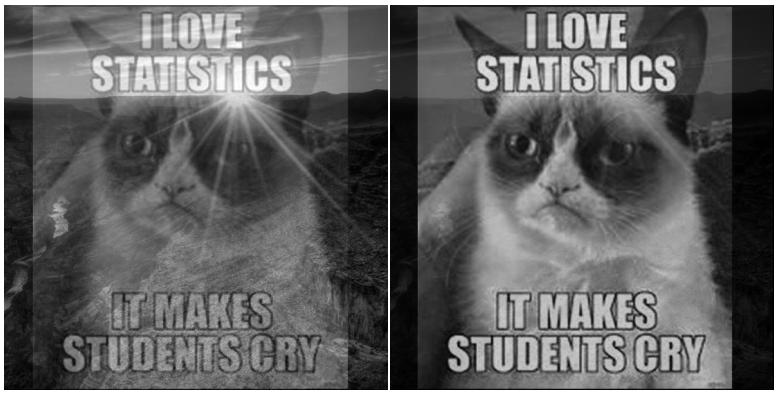

Figure 2: Mixed images for two-image test.

```
## Construct the mixture data frame
S1 <- cbind(img1_vec, img2_vec)
```

Here, we first overlay two images by simply add them together. To adjust the proportion of each image in the mixture, we simple multiply the image by a factor. Then we turn the matrices into vectors, and then combine the vectors into a dataframe.

5. Perform ICA and check results.

```
# FastICA
set.seed(1)
ICA1=fastICA(S1, n.comp = 2, # Number of components to be extracted
             alg.typ = "parallel", # Here, alg.typ = "parallel"
             #indicates that the components are extracted simultaneously

             fun = "logcosh", # the functional form (log cosh here)
             #of the G function used in the approximation to neg-entropy.

             alpha = 1, # the value associated with logcosh function
             method = "C", # use C code to perform most of the computations.
             row.norm = FALSE,
             maxit = 200, # Max # of iterations
             tol = 0.0001, # covergence tolerance
             verbose = TRUE)
```

```
## Centering
## Whitening
## Symmetric FastICA using logcosh approx. to neg-entropy function
## Iteration 1 tol=0.455420
## Iteration 2 tol=0.131201
## Iteration 3 tol=0.010040
## Iteration 4 tol=0.000438
## Iteration 5 tol=0.000019
```

```
## Reconstruct image by converting vectors back to 600*600 matrices.
img1_ica=as.Image(matrix(ICA1$S[,1], nrow=600))
img2_ica=as.Image(matrix(ICA1$S[,2], nrow=600))

ica1_img=combine(img1, img1_ica, test2,
                 img2,img2_ica, test1) # Combine images

plot(ica1_img, all=TRUE) # Plot all

names=c("Mixture", "ICA", "Source") # Make labels
for(i in 1:3){ # Label each panel
  name=names[i]
  for(j in 1:2){
    text(x=600*(i-1)+20, y=600*(j-1)+20, label=paste(name, " ", j, sep = ""),
         cex=1, adj = c(0,1), col = "red")
  }
}
```

Here we apply `fastICA` algorithm on the two image mixtures (in dataframe form), and the output is a list

Figure 3: A comparison of mixed images, ICA components and source images for two-image test.

of objects containing the ICA components, which are the estimated source images. The estimated sources are in the dataframe format with the same dimension as the input. Then we can "reconstruct" the ICA components by converting the vectors back into 600×600 matrices, and then plot them. The discussion of the ICA components recovered from mixtures will be included in the Discussion part.

6. An example of non-linear combination.

```
### A nonlinear transformation test.
w = makeBrush(size = 21, shape = 'gaussian', sigma = 5) # Create brush
t2.f=filter2(test2, w); #display(t2.f) # Blur image 2 to create a non-linear transformation

img2.nl=0.2*test1+0.8*t2.f
plot(combine(img1, img2.nl), all=T)
```

Here we have applied a blur filter onto the test image 2 (Grand Canyon) and use this to create **one of the mixtures** (the other mixture is the same as we had previously used). We treat this as a non-linear transformation of the source. Then we want to test the performance of ICA.

```
img1_vec=as.vector(img1) # Convert to vector, default is by row
img2_vec.nl=as.vector(img2.nl)

S2 <- cbind(img1_vec, img2_vec.nl)
set.seed(2017)
ICA2=fastICA(S2, 2,  alg.typ = "parallel", fun = "logcosh", alpha = 1,
          method = "C", row.norm = FALSE, maxit = 200,
```

Figure 4: Mixed images for non-linear transformation/combination test.

```
          tol = 0.0001, verbose = TRUE)
```

```
## Centering
## Whitening
## Symmetric FastICA using logcosh approx. to neg-entropy function
## Iteration 1 tol=0.234511
## Iteration 2 tol=0.028225
## Iteration 3 tol=0.004681
## Iteration 4 tol=0.000560
## Iteration 5 tol=0.000076
```

```
img1_ica2=as.Image(matrix(ICA2$S[,1], nrow=600))
img2_ica2=as.Image(matrix(ICA2$S[,2], nrow=600))

ica1_img=combine(img1, img1_ica2, test2,
                 img2.nl,img2_ica2, test1) # Combine images
plot(ica1_img, all=TRUE) # Plot all
for(i in 1:3){ # Label each panel
  name=names[i]
  for(j in 1:2){
    text(x=600*(i-1)+20, y=600*(j-1)+20, label=paste(name, " ", j, sep = ""),
         cex=1, adj = c(0,1), col = "red")
  }
}
```

The results are worse than the first case.

7. Test of mixtures from three images.

```
### A linear transformation of test1, 3 and 4
img1=0.6*test1+0.2*test3+0.2*test4
img2=0.2*test1+0.7*test3+0.1*test4
img3=0.1*test1+0.2*test3+0.7*test4
plot(combine(img1, img2, img3), all=T)
```

```
## Convert the two transformations to vectors
img1_vec=as.vector(img1)
```

Figure 5: A comparison of mixed images, ICA components and source images in the non-linear transformation test.

Figure 6: Mixed images from three sources.

```
img2_vec=as.vector(img2)
img3_vec=as.vector(img3)

## Construct the mixture data frame
S3 <- cbind(img1_vec, img2_vec, img3_vec)
```

```
# FastICA
set.seed(2017)
ICA3=fastICA(S3, 3,  alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "C", row.norm = FALSE, maxit = 200,
             tol = 0.0001, verbose = TRUE)
```

```
## Centering
## Whitening
## Symmetric FastICA using logcosh approx. to neg-entropy function
## Iteration 1 tol=0.165148
## Iteration 2 tol=0.001039
## Iteration 3 tol=0.000012
```

```
img1_ica3=as.Image(matrix(ICA3$S[,1], nrow=600)) # Reconstruct image
img2_ica3=as.Image(matrix(ICA3$S[,2], nrow=600))
img3_ica3=as.Image(matrix(ICA3$S[,3], nrow=600))

ica3_img=combine(img1, img2, img3,
                 img1_ica3, img2_ica3, img3_ica3,
                 test3, test1, test4)
plot(ica3_img, all=TRUE)
for(i in 1:3){
  name=names[i]
  for(j in 1:3){
    text(x=600*(j-1)+20, y=600*(i-1)+20, label=paste(name, " ", j, sep = ""),
         cex=1, adj = c(0,1), col = "red")
  }
}
```

8. Discussion

There are some observations that I found interesting:

- Other than the non-linear case, ICA did a great job in identifying the source images. But we did observe some distortions to the images, such as higher contrast and loss of details.
- The assumption of linear combination of sources is important for ICA to perform well. In our non-linear combination case, the component that ICA identified as "Grand Canyon" looks like the average of the original image and the blurred version of it (Fig. 6, "ICA 1"). Also comparing with Fig. 3, we saw more "spillover" of one source image to another.
- ICA seems to provide a **relative** solution, but not the unique solution. In Fig.3 we found that ICA 1 component has the right relative information of the image (contrasts and details), but the overall color seems to be inversed. And the `fastICA` algorithm has randomness in it as the overall color depends on the seed you set.
- In terms of the loss of details, it is possible that when the colors of some parts of the source images are very similar, ICA will not be able to separate them very well.

Figure 7: A comparison of mixed images, ICA components and source images from the three-image test.