Faculty of Applied Sciences

High-dimensional data analysis

MATH2021-1

Project 3:
Independent Component Analysis

Authors :
Delcour Florian - s181063
Makedonsky Aliocha - s171197

Instructors :
Haesbroeck Gentiane

December 2021

1 Explanation of arguments of `fastICA`

To realize this project, we used the function `fastICA()` from the package `fastICA`.
This function has several arguments that we will list below :

- `X` : matrix with our data, each column represents a variable, so an image in our case. We used `image_tot` as variable for this argument, which was created by transforming the different 300x300 matrices representing the given images, into one column vector for each, of 90 000 elements, and then we bind those 3 vectors into a matrix of dimensions 90 000x3, one column for each image.
- `n.comp` : the number of components that the function needs to extract. Here the value is 3 since we have 3 components, one image being one component.
- `alg.typ` : can take either the value `"parallel"` to extract the components simultaneously, either the value `"deflation"` to extract the components one by one. Having tried both values, we preferred `"deflation"`.
- `fun` : function used to make the approximation of the neg-entropy. Can take the value `"logcosh"` ($G(t) = \frac{1}{\alpha} \log \cosh \alpha t$ with $1 \leq \alpha \leq 2$) to use the log of the cosh or `"exp"` ($G(t) = -\exp(-t^2/2)$) to use the exponential, both were presented in the lectures as they do not grow too fast so they provide robustness. Having tried both values, the results were quite similar, the colors are a bit more marked with `"logcosh"` for the un-mixed image 3 so we chose it.
- `alpha` : only used if we choose `"logcosh"` for the `fun` argument, seeing the expression of the logcosh function in the previous argument. We have tried several values for α, in the majority of the cases the results were similar, but slightly more differentiable with $\alpha = 1$ so we took this value.
- `method` : can take the value `"R"` to compute everything in R, or the value `"C"` to compute with C code, which is faster, so we chose `"C"`.
- `row.norm` : logical value to standardize or not the rows of the data matrix, which is useless since they are already values between -1 and 1 because it's just values of grey.
- `maxit` : maximum number of ICA iterations, we set it high, to 200, but the number of iterations needed is proportional to the number of components, which is low here so we will never reach 200 iterations.
- `tol` : tolerance under which we stop iterating ICA because the accuracy is enough. We can set this tolerance pretty low (1e-7) since the number of components to extract is low too, so a lot of iterations can be performed quickly to improve accuracy.
- `verbose` : logical value to have info on the computation of the algorithm, we set it to `TRUE`.
- `w.init` : initial un-mixing matrix, we let it to default so a matrix of normal random variables is used.

2 `fastICA` algorithm

The data is 1 value for each pixel, representing the intensity of grey color of the different pixels composing the image.

We need to pre-process the data (the mixture), which is `X` in the arguments we spoke about in the previous section, that we called `image_tot` in our code.

Firstly, we need to substract each column of X by the mean of this same column, so now the mixture variables all have a zero mean. This will also center the future estimated sources. It will simplify the computation.

Then, we must whiten the data to have components that are uncorrelated and with unit variance. To whiten them, we must project them on the 3 principal components (the number of different variables so 3 here, because we have 3 images) by multiplying `X` by `K` which is a pre-whitening matrix. This matrix `K` is obtained by computing the inverse of the square root of the covariance matrix, so we have $X \to KX = \Sigma^{-1/2}X$

The number of mixtures being equal to the number of sources (we have 3 mixtures, which are the 3 images that we were given, and we have 3 sources, which are the 3 original pictures that we must estimate) so there is no need to perform any dimension reduction, and we don't have either an "over-complete" problem.

Finally, the un-mixing matrix `W` is chosen in order to minimize the neg-entropy `J`, which is expressed as $J(y) = [E\{G(y)\} - E\{G(v)\}]^2$ where $v \sim \mathcal{N}(0,1)$, y is our different values of pixel, and G is the function used by the argument `fun` presented in the previous section, so $G(y) = \dfrac{1}{\alpha} \log \cosh \alpha y$ with $1 \leq \alpha \leq 2$ here.

Here are the 3 mixed images we were given :

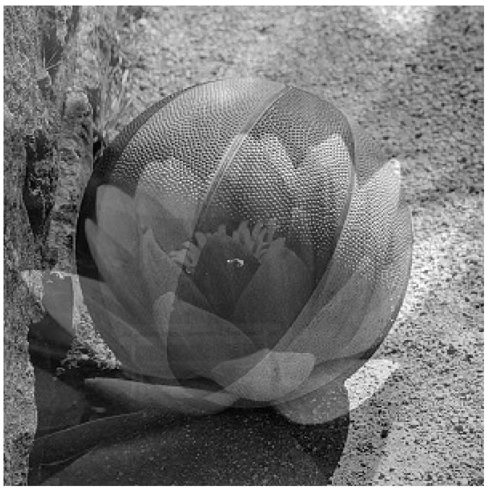
Figure 2: Mixed image 2

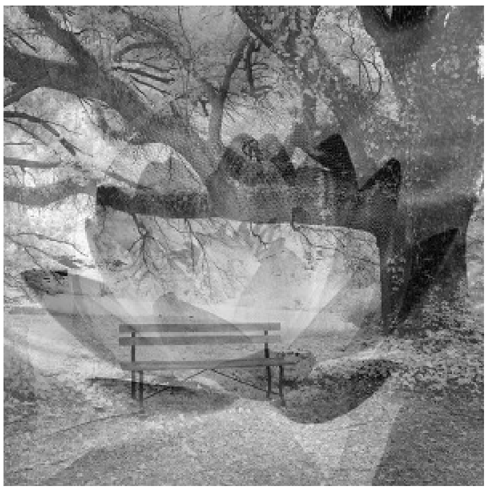
Figure 3: Mixed image 3

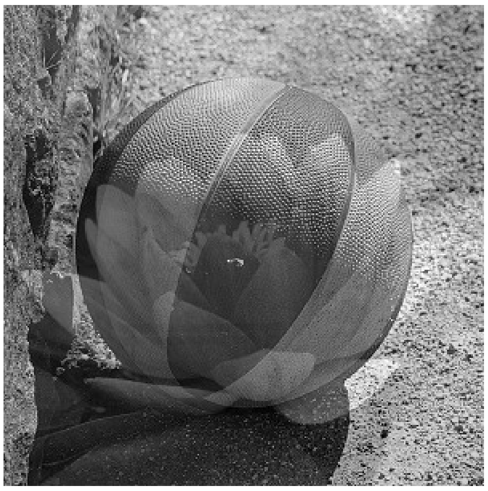
Figure 1: Mixed image 1

The estimated mixing matrix is given by :

$$A = \begin{pmatrix} 0.21265542 & 0.186964266 & -0.08390278 \\ 0.04274506 & 0.077501063 & -0.11283565 \\ 0.02009655 & 0.004117959 & 0.10683383 \end{pmatrix}$$

We observe quite small values, close from 0 for certain, that's not a good sign for the ICA, because it doesn't yield clear directions.

Here are the 3 images we obtain :

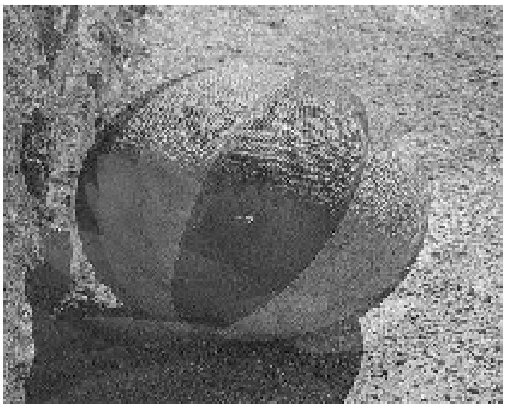
Figure 4: Estimated image 1

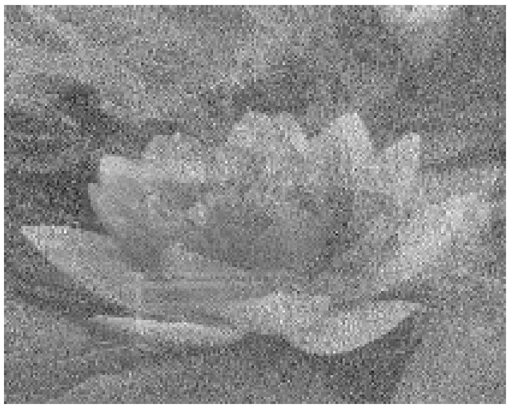
Figure 5: Estimated image 2

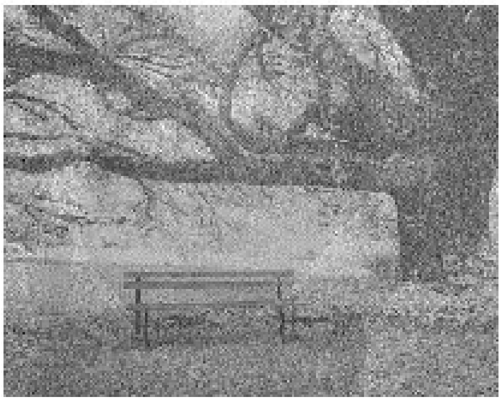
Figure 6: Estimated image 3

Obviously we see that they aren't perfect, it's normal since ICA just does an estimation of the sources so it couldn't be perfect. But some factors may explain this non-perfection.
Our image 1 corresponds to the image_1 on ecampus, our image 2 corresponds to the image_14 on ecampus and our image 3 corresponds to the image_3 on ecampus. In the `.zip` file we provided, we denoted our estimated images by `estimated_image_`, the real images by `real_image_` and the mixed images we were given by `DelcourMakedonsky_Mix`. Here are the real ones :

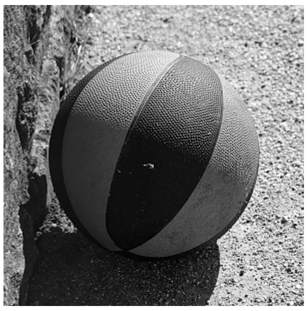
Figure 7: Image 1 on ecampus corresponding to our image 1

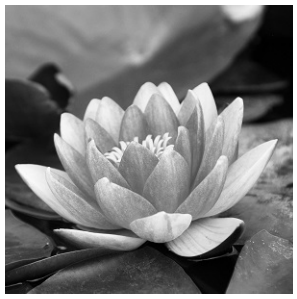
Figure 8: Image 14 on ecampus corresponding to our image 2

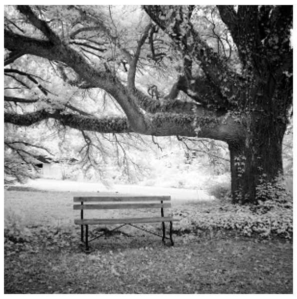
Figure 9: Image 3 on ecampus corresponding to our image 3

3 Factors influencing quality of ICA

3.1 Independence of the sources

The sources must be independent to perform ICA, or it would be too complex to separate the different sources from the mixtures. A measure of independence is correlation, though un-correlation is not sufficient to prove dependence but it's still an easily observable measure.

We see on Figure 10 that the estimated sources are totally uncorrelated, so there should

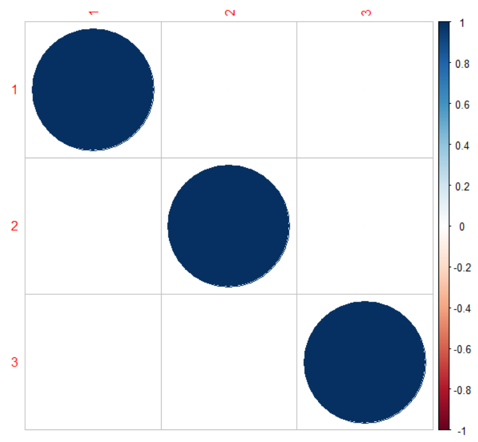

Figure 10: Correlation matrix

be no problem from this side. The covariance matrix is the same, it's a diagonal matrix with 1 on the diagonal.

3.2 Gaussianity of the sources

If the sources are gaussian, i.e. they follow a normal distribution, the observed data, once plotted, would look like a circle, so there wouldn't be any observable direction available for the projection. Let's plot an histogram for each of the estimated sources.

We see that the sources are gaussian, which is not good at all for ICA. That's probably what explains the non-perfection of the estimated sources, that's why we don't exactly recover the initial images.

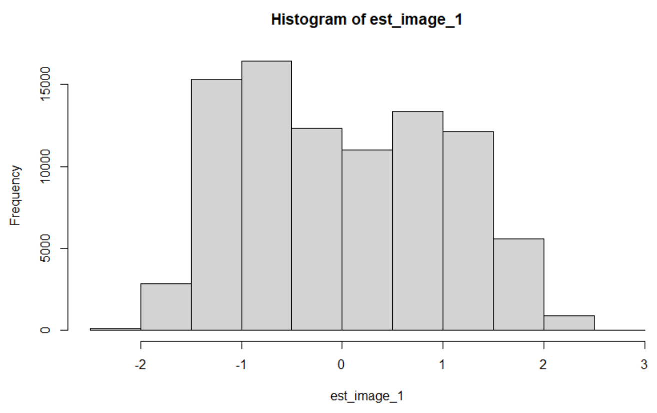

Figure 11: Histogram of the estimated image 1

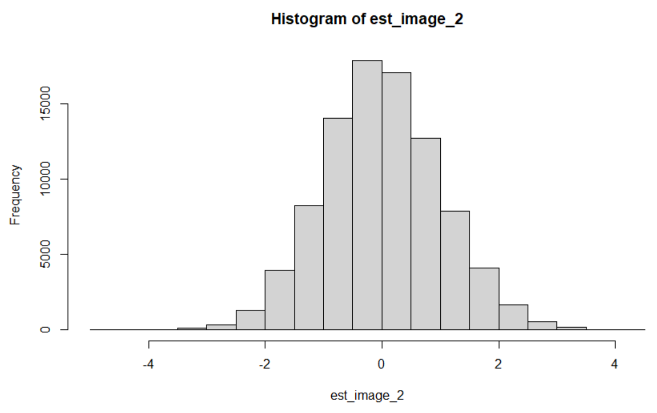

Figure 12: Histogram of the estimated image 2

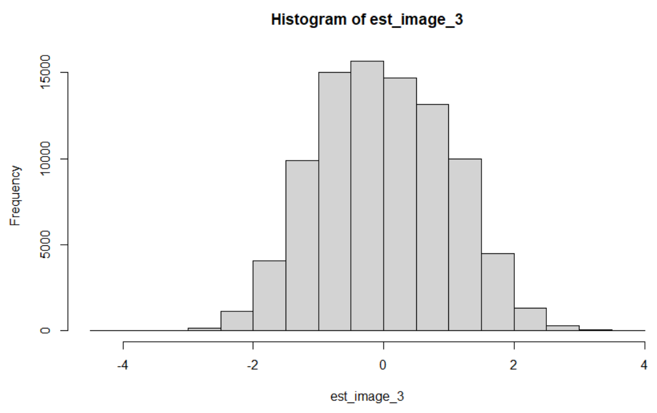

Figure 13: Histogram of the estimated image 3