## APPLIED SIGNAL PROCESSING 2017

## EXERCISE 3

### a) Image compression using PCA

Consider an $N_\mathrm{r} \times N_\mathrm{c}$ array $\mathbf{X}(m,n)$ representing an image. For data compression, represent the image data in the form of a data matrix $\mathbf{W}(i,j)$ (see the comments below for possible ways to define $\mathbf{W}(i,j)$), and compute the singular value decomposition

$$\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathrm{T} = \sum_{i=1}^{p} \sigma_i \mathbf{u}_i \mathbf{v}_i^\mathrm{T}$$

Plot the singular values $\sigma_i$. Introduce next a compressed approximation $\mathbf{W}_r$ of $\mathbf{W}$ using only the $r$ first dominant principal components,

$$\mathbf{W}_r = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\mathrm{T} = \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^\mathrm{T}$$

where $\mathbf{U}_r = [\mathbf{u}_1 \cdots \mathbf{u}_r]$, $\mathbf{\Sigma}_r = \mathrm{diag}\,(\sigma_1, \ldots, \sigma_r)$ and $\mathbf{V}_r = [\mathbf{v}_1 \cdots \mathbf{v}_r]$. Next reconstruct the compressed image $\mathbf{X}_r(m,n)$ corresponding to compressed data matrix $\mathbf{W}_r(i,j)$, and compare with the original.

Determine a suitable value for $r$ which gives a compressed image of reasonable quality, and calculate the amount of data needed to represent the compressed image. Compute also the average root-mean-square approximation error

$$e(\mathbf{W} - \mathbf{W}_r) = \Big( \sum_{i,j} \frac{1}{N_\mathrm{r} N_\mathrm{c}} \big(\mathbf{W}(i,j) - \mathbf{W}_r(i,j)\big)^2 \Big)^{1/2}$$

and determine the relative root-mean-square approximation error $e(\mathbf{W} - \mathbf{W}_r)/e(\mathbf{W})$.

**Comments:**

The image can be selected freely, or taken as one of the images in Exercise 2b.

The data matrix $\mathbf{W}$ can be defined as either
- the image itself, $\mathbf{W} = \mathbf{X}$, or
- taking the columns of $\mathbf{W}$ as the 64 elements of non-overlapping $8 \times 8$ blocks of $\mathbf{X}$.

The second method usually gives better compression.

The following Matlab commands may be useful: the Matlab command `a=A(:)` forms a vector $\mathbf{a}$ from the columns of the matrix $\mathbf{A}$. The matrix $\mathbf{A}$ can be reconstructed using the command `A=reshape(a,M,N)`, where $M$ and $N$ are the number of rows and columns of $\mathbf{A}$.

Note that better compression is obtained by defining the data matrix $\mathbf{W}$ in such a way that it has zero mean value, i.e., by subtracting the mean

$$X_\mathrm{mean} = \frac{1}{N_\mathrm{r} N_\mathrm{c}} \sum_{m=1}^{N_\mathrm{r}} \sum_{n=1}^{N_\mathrm{c}} \mathbf{X}(m,n)$$

from all elements and adding it back at reconstruction. (The mean value $X_{\text{mean}}$ is a scalar quantity, whereas the singular value decomposition of a matrix with all elements equal to $X_{\text{mean}}$ would require two singular vectors.)

**b) Blind source signal separation using ICA**

**(i)** The files `source1.dat` and `source2.dat` contain 8192 Hz audio signals $y_1(n)$ and $y_2(n)$. Construct two mixed signals $w_1(n)$ and $w_2(n)$ according to

$$\begin{bmatrix} w_1(n) \\ w_2(n) \end{bmatrix} = \mathbf{A} \begin{bmatrix} y_1(n) \\ y_2(n) \end{bmatrix} \quad n = 0, 1, \ldots, N-1$$

where $\mathbf{A}$ is a randomly selected 2-by-2 matrix. Then use the FastICA algorithm to unmix the signals $w_1(n), w_2(n)$ to find independent signal components $s_1(n), s_2(n)$. Compare the calculated independent components with the original signals $y_1(n)$, $y_2(n)$ by plotting the original signals and the reconstructed signal in the range $n =$ 20000–20100. See Remark regarding the need for scaling and possible sign inversion.

**(ii)** Repeat the source signal separation in (i) by using the first 10000 data points only to find the reconstruction matrix $\mathbf{B}$, such that $\mathbf{s}(n) = \mathbf{B}\mathbf{w}(n)$, and reconstruct the independent components for the whole sequence using the identified matrix $\mathbf{B}$.

**(iii)** The files `w1.dat` and `w2.dat` contain 11025 Hz audio signals $w_1(n)$ and $w_2(n)$, which consist of two mixtures of a sound signal and a heavy noise signal (such as that from a vacuum cleaner). Use ICA for extracting the sound signal buried in the noise. Verify that the sound signal has been found by listening to it (note the sampling frequency!), and by comparing the result with the original signal, which is given in the file `sound50000to50100.dat` for the range $n =$ 50000–50100. Note again, that scaling and possible sign inversion is needed (cf. Remark).

**Remarks:**

The calculations can be performed using the FastICA program package, which can be downloaded from the web.

Usage: `[IC,A,B] = fastica(W,OPTIONS)`, where

`W`: signal matrix with columns $\mathbf{w}(n)$,

`IC`: calculated source signal matrix with columns $\mathbf{s}(n)$,

`A,B`: calculated mixing and reconstruction matrices; $\mathbf{w}(n) = \mathbf{A}\mathbf{s}(n), \mathbf{s}(n) = \mathbf{B}\mathbf{w}(n)$.

Various optional parameters controlling the algorithm can be set with the OPTIONS input.

Observe that FastICA determines the independent components $s_i(n)$ in arbitrary order, the components are normalized to have unit variance (whereas there is no such assumption on the original signals $y_i(n)$), and they are arbitrary with respect to sign ($-s_i(n)$ and $s_i(n)$ are equally valid independent components). For comparison, $y_i(n)$ and $s_i(n)$ should therefore be scaled to have equal variances, and possible sign differences should observed.