EX5: ICA and wavelets

General remarks

Your MATLAB scripts (one per exercise) should produce all the desired plots with one click of the 'run'-button, without further intervention! (Make sure all plots for an exercise stay available together, after the script(s) have been executed.) Document your code to facilitate debugging, reuse, and evaluation (in case of problems).

Evaluation

For this session, you should write a report, which will be discussed at the oral exam. Please upload your full code and report on Toledo at the latest on **December 20, 23:59**. Do not include data and packages or scripts we provided, but only the files you wrote or modified. Also, a printed version of your report should be delivered at the ESAT reception by the same date before **14:00**.

1 Independent component analysis

Background

Independent component analysis (ICA) is a so-called blind source separation (BSS) technique. In the BSS problem a multi-channel (e.g., different sensors) signal $\mathbf{x}[t] = \begin{bmatrix} x_1[t] & x_2[t] & \cdots & x_N[t] \end{bmatrix}^{\mathrm{T}}$ is observed, with N the number of channels and $t = 1 \dots T$ the time samples. $\mathbf{x}[t]$ is the result of an unknown linear mixture of a set of M source signals $\mathbf{s}[t] = \begin{bmatrix} s_1[t] & s_2[t] & \cdots & s_M[t] \end{bmatrix}^{\mathrm{T}}$, with $t = 1 \dots T$.

This can be mathematically modeled via the unknown mixing matrix $\mathbf{A}$:

$$\underbrace{\mathbf{x}}_{\text{known}}[t] = \underbrace{\mathbf{A}}_{\text{unknown}}\underbrace{\mathbf{s}}_{\text{unknown}}[t]$$

This is called a 'blind' source separation problem, as both the mixing matrix $\mathbf{A}$ and source signals $\mathbf{s}[t]$ are unknown. Note that both mixtures and sources are assumed to be zero-mean, which can be easily obtained by a priori centering the data.

The goal is to estimate a *demixing* matrix $\mathbf{W}$ (or equivalently: its (pseudo-)inverse, which corresponds to the mixing matrix $\mathbf{A}$) such that

$$\mathbf{y}[t] = \mathbf{W}\mathbf{x}[t]$$

is equal to the unknown source signals $\mathbf{s}[t]$, up to scaling and permutation of the sources. Without assumptions, it is, in general, impossible to solve this problem. ICA resolves this by assuming and imposing statistically independence between the source signals $s_1[t], \dots, s_m[t]$, in order to retrieve the sources. Read [1] as background material.

The objectives of this exercise are:

- to explore and analyze the different properties of ICA and the fastICA algorithm.
- To learn how to use PCA as a preprocessing step before ICA.
- To apply ICA on EEG data to remove eye-blink artifacts.

Programming exercises

1.1 Exploration and analysis of ICA (properties) on a mixture from ECG

In this exercise, four sources are being used, each of 10 s long and sampled at 500 Hz. The first and third source are physiological signals, resp. an ECG and PPG signal from PhysioNet [2]. The second source (a sawtooth - representing, e.g., instrumentation noise) and fourth source (white Gaussian noise) simulate noise sources. Imagine we had a way to simultaneously measure different mixtures (e.g., with different sensors) of these four signal components. This is simulated by creating different mixtures of the sources.

Open `ex5_1_1.m`, which is included in the assignment on Toledo and which provides the skeleton for this exercise. The data in `data_ex5_1_1.mat` is automatically loaded: `S` contains the source signals, while `A` and `A2` contain the mixing matrices. First, normalize the sources using the `zscore`-function such that their mean value is zero and they have unit variance. The sources are then displayed using the script. Complete the script based on the tasks below.

1.1.1 Measuring four mixtures

The four sources are now being mixed into four signals (measured with, e.g., four sensors) $\mathbf{x}[t]$ using the given mixing matrix $\mathbf{A}$ (this has already been completed in the script). Display these mixtures. Now perform the following tasks.

a. Demixing the mixtures using fastICA
Apply the fastICA algorithm to the mixtures $\mathbf{x}[t]$ using the provided `fastica`-function to estimate the sources.

b. Normalization of the estimated sources
Normalize all the estimated sources such that their mean value is zero and they have unit variance.
Display the resulting normalized sources.

c. Identification of the estimated sources
Automatically match the estimated sources with the corresponding original sources (the ECG signal with its estimate, Gaussian noise with the appropriate estimate, ...) using correlation (`corr`) as the similarity metric. Plot every estimate and corresponding matched source next to each other and visually check your pairing procedure. Make sure the plots are intelligible.

d. Quality assessment of the estimated sources
Calculate the root-mean-square error (RMSE) per matched pair of estimated source and original source:

$$\mathrm{RMSE_m} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(y_m[t] - s_m[t]\right)^2},$$

with T the number of time samples, $y_m[t]$ the m^{th} estimated source and $s_m[t]$ the m^{th} original source signal. When computing the RMSE values, make sure that the estimated sources have the same sign as the original sources (do this in automatic way).

e. Artifact removal using ICA
As the white Gaussian noise and sawtooth signal represent (instrumentation) noise, remove these noise sources from the measured mixtures $\mathbf{x}[t]$ in the sensor space. Remove these artifacts by modifying the mixing matrix. Plot the corresponding mixture signals $\mathbf{x}[t]$ before and after removing the noise signals. Pay extra attention to the normalization of the estimated sources: should they be normalized?

1.1.2 Measuring 30 mixtures

Imagine now that we have much more sensors at our disposal and simultaneously measure 30 mixtures of these four sources.

a. fastICA without additional noise
Apply the fastICA algorithm to the mixtures $\mathbf{x}_2[t]$, produced using the given mixing matrix $\mathbf{A}_2$ (this has already been completed in the script). Match the (estimated) sources, plot the estimated and original sources again next to each other and compute the RMSE values. Is the algorithm able to identify the number of sources? How does it work? To support your analysis, check the preprocessing steps of the fastICA algorithm and check the output information of the fastICA algorithm. Analyze the eigenvalues of the covariance matrix to show that you understand the underlying mechanisms.

b. fastICA with additional noise
Repeat the above for the third mixture $\mathbf{x}_3[t]$, which is equal to $\mathbf{x}_2[t]$ with additional noise (uniformly distributed) added to the mixed signals after mixing the four sources (this has already been completed in the script). Is the algorithm able to identify the number of sources? Why (not)? Visually check the estimated sources and analyze the RMSE values and compare with the previous RMSE values. What do you conclude now? Link this again with the preprocessing steps and the output of the fastICA algorithm. What is the difference in the output information displayed, compared to before?

c. Dimensionality reduction as a preprocessing step for ICA
Improve on the previous source separation results, based on mixture $\mathbf{x}_3[t]$, by using principal component analysis (PCA) as a preprocessing step (see the `pca`-command). Plot the eigenvalues of the covariance matrix to make a principled choice of the number of components retained and link this to your knowledge about the BSS problem. Reduce the dimensionality of the mixture $\mathbf{x}_3[t]$ using PCA and reapply the fastICA algorithm on the new mixtures. Again, match and plot the obtained estimates and corresponding sources. Compute the RMSE values and compare with the previous results.

1.1.3 Additional questions

Answer, based on your plots and the tasks completed in this exercise, the following additional questions (number them in your report as they are listed here):

1. In 1.1.1b, you normalized the estimated sources. Why is this important? Is this necessary for ICA?

2. In 1.1.1c, you matched the estimated sources with the original sources. Why do we need to do this? Could we not directly use the output of the fastICA-algorithm?

3. When computing the RMSE values in 1.1.1d, you sometimes needed to change the sign of the estimated sources. Why is this?

4. Should the estimated sources be normalized in 1.1.1e? Why (not)?
5. Is the fastICA algorithm able to identify the number of sources in 1.1.2a? How does it work?
6. Is the fastICA algorithm able to identify the number of sources in 1.1.2b? Why (not)?
7. Suppose an additional Gaussian noise source is present. Would fastICA be able to separate both of them? Why (not)?

1.1.4 Deliverables

- Completed implementation of `ex5_1_1.m`.
- Plots of the original sources next to their best-matching components extracted from the mixtures, for all scenarios. Mention and discuss the RMSE values.
- Plots of the mixtures before and after removal of the sawtooth and Gaussian noise source, side-by-side.
- Analysis of the output (information) and preprocessing of the fastICA algorithm in the case of 1.1.2a and 1.1.2b, as well as of the eigenvalues of the covariance matrices (integrate this in your answer to question 5 and 6).
- Plot of the eigenvalues of the covariance matrice when measuring 30 mixtures with noise (1.1.2c) and corresponding analysis in the context of PCA.
- Discussion of all your results and plots.
- Check that you have answered all additional questions raised in Section 1.1.3 and make sure that they can be easily found back in your report.

1.2 ICA to remove eye blink artifacts from EEG

In this exercise, 20 s of EEG data of an auditory attention experiment is provided, in which participants focus attention on one of two speech signals. The EEG data is measured using twelve channels, at sampling frequency 128 Hz. The data is first loaded into the variables `eeg` (the EEG data), `channels` (the names of the channels) and `fs` (the sampling frequency). Figure 1 shows the locations of the different channels. Complete the script `ex5_1_2.m` based on the tasks below, in order to remove the eye blink artifacts from the EEG, as they do not contain relevant information for the application at hand.

a. Visualization of the data
Visualize the data using the function `eegPlot.m`. Do you notice eye blinks? Indicate them in the plot.

b. Demixing the EEG and identifying the eye blinks
Use the fastICA algorithm to demix the EEG. Look at the temporal and spatial information. The spatial information can be obtained using `topoPlot.m`, of which an example is shown in Figure 1.

c. Removal of the eye blink artifacts
Remove the eye blink artifacts similarly as you did with the sawtooth and the Gaussian noise in the previous exercise. Visualize the result.

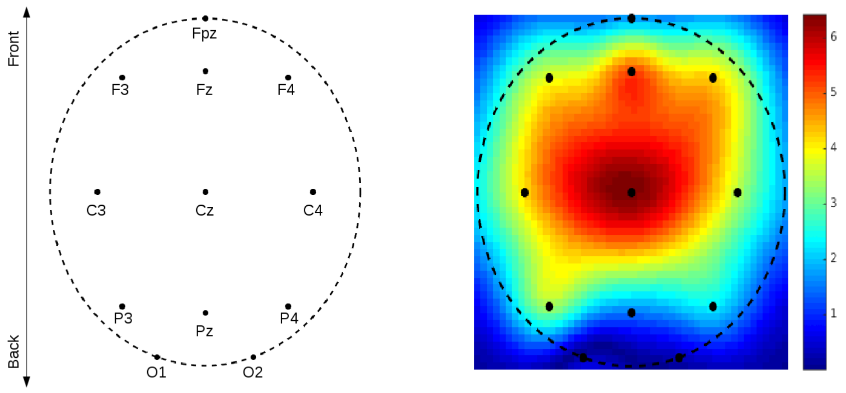


Figure 1: The EEG configuration (left) and an example of a topographic plot (right).

1.2.1 Additional questions

Answer, based on your plots and the tasks completed in this exercise, the following additional question:

1. In what EEG channels are the eye blinks best noticeable? Why?

1.2.2 Deliverables

- Completed implementation of `ex5_1_2.m`.
- Discussion of all your results and answer to the question in Section 1.2.1.
- Plots of the original EEG with the indicated eye blinks.
- Plots illustrating the selection of the artifact components (temporal and spatial).
- Plot of the clean EEG after artifact removal, clearly showing the change in comparison to the original signal.

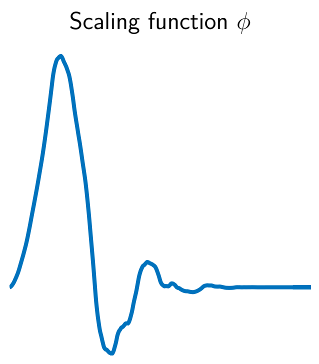
2 Wavelets

Background

The Fourier transform or related approaches such as the Gabor transform are very effective when studying signals composed of smooth oscillations. However, they analyze all frequency regions with the same resolution, which makes them less suitable, e.g., when signals possess sharp transients and discontinuities. For such purposes, wavelets are more suitable, as they allow to localize higher frequencies more precisely in time and allow to analyze slow-varying components more precisely in the frequency domain. This is why wavelets are often used in, for example, ECG signal analysis, as the R-peak is a sharp transient.

The objectives of this exercise are:

- getting familiar with wavelets.
- Compression of an ECG signal, using wavelets.
- Epileptic seizure detection in EEG, using wavelets.



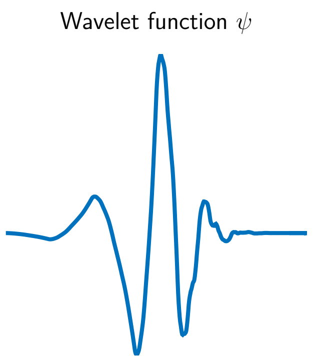


Figure 2: The scaling and wavelet function of the Daubechies wavelet of order four.

Programming exercises

2.1 ECG signal compression using wavelets

In this exercise, 15 s of a single-lead ECG measurement is provided, sampled at 300 Hz. To store this ECG signal in the time domain, 4500 samples are needed. However, by using a wavelet decomposition, we can transform this ECG signal into another domain and reconstruct it using fewer coefficients, each of them corresponding to certain components of the ECG signal, whilst losing a minimal amount of information (signal components). Complete `ex5_2_1.m` based on the task below.

a. The wavelet decomposition of the ECG signal
Decompose the original ECG signal with the Daubechies wavelet of order four (`db4`, see Figure 2), using five levels of decomposition. The scaling function is related to the low-pass filter in the hierarchical filterbank decomposition and computes the so-called 'approximation coefficients' in each filtering stage (i.e., the low-frequency part). The wavelet function is related to the high-pass filter that computes the 'detail coefficients' in all stages (i.e., the high-frequency part). Plot the approximation and detail coefficients in the following specific order: (A5, D5, D4, D3, D2, D1), using subplots (stacked below each other) and add a subplot on top with the time-domain ECG signal (so seven subplots in total). Make sure that the wavelet coefficients are aligned correctly with the corresponding time points in the ECG signal above. Study and use the functions `wavedec` and `waverec`, `appcoef` and `detcoef`. For plotting, use the `subplot`- and `stem`-functions.

b. Compression of the ECG signal using 10 *coefficients*
Of all wavelet coefficients (approximation and detail coefficients), keep the $M = 10$ coefficients with the largest *absolute* values (10 in total across all levels). Set all the remaining coefficients to zero and reconstruct the ECG signal. Plot the kept coefficients (`stem`) as in the previous question. Plot the results of the reconstructed signal and compare it visually with the ECG (plot on the same subplot in different colors).

c. Compression of the ECG signal using $25, 50$ *and* 100 *coefficients*
Repeat the above whilst keeping $M = 25, 50, 100$ coefficients respectively. Similarly as before, plot the retained coefficients and reconstructed signals along with the original ECG. Comment on the obtained results. Calculate the RMSE for each case, as well as the compression rate (CR) as follows:

$$\text{CR} = \frac{\text{uncompressed size}}{\text{compressed size}},$$

where the uncompressed signal corresponds to the original signal in time domain and the compressed signal with the compressed wavelet coefficients. Compare and link the RMSE values with the amount of compression.

2.1.1 Additional questions

Answer, based on your plots and the tasks completed in this exercise, the following additional questions (number them in your report as they are listed here):

1. Given the wavelet decomposition in 2.1a, which coefficients (A5, D5, D4, D3, D2, D1) would you use to analyze the QRS complexes? Knowing that the sampling frequency is 300 Hz, what is the corresponding frequency range?

2. Compare the reconstructed ECG signal with 10 coefficients from 2.1b with the shapes of the scaling ϕ and the wavelet ψ functions from Figure 2. Why can you recognize ϕ at some places of the reconstructed ECG signal and ψ at other places?

3. In this exercise, the Daubechies wavelet of order four has been used. Why would you prefer one wavelet over another?

2.1.2 Deliverables

- Completed implementation of `ex5_2_1.m`.
- For all values of M $(10, 25, 50$ and $100)$, provide a series of subplots, as well as for the original wavelet decomposition. The original ECG signal on top, overlaid with the reconstructed ECG signal. Then, below, provide `stem` plots showing the selected approximation and detail coefficients, each of them in a separate subplot in the aforementioned order. Don't forget to mention and discuss the RMSE and CR values.
- Discussion of all your results and plots.
- Check that you have answered all additional questions raised in Section 2.1.1 and make sure that they can be easily found back in your report.

2.2 Epileptic seizure detection

We saw that wavelets provide an easy way to characterize the shape of a signal. Hence, it was useful for signal compression in the previous exercise. We can also view the discrete wavelet transform as a filterbank of bandpass filters. This is particularly useful if we want to study a signal in several non-uniform frequency bands. We will now use this interpretation to classify EEG segments containing epileptic seizure activity.

In this exercise, (noisy) single-channel EEG data, sampled at 256 Hz, are being used. The EEG data have already been segmented in (partly overlapping) segments of 2 s. In the `ex5_2_2.m`, the EEG data are first loaded, containing `dataTrain` and `dataTest`, which consist of 124/142 segments of 2 s. The corresponding variables `labelsTrain` and `labelsTest` indicate whether a segment contains seizure activity (label 1) or not (label 0). Complete the script based on the tasks below.

a. Data exploration
Visualize your data by plotting all non-seizure segments on top of each other in one plot, with the seizure segments below it in another plot. What discriminant property do you observe? How would you quantify this?

Furthermore, do you think we should normalize/standardize the raw data? If so, add this to the script.

b. Full signal energy as a feature
Signal energy is one way to quantify the difference between seizure/non-seizure activity. Given a discrete-time signal $s[n]$, $n = 1 \dots T$, its energy is defined as the sum of the squared samples:

$$E(s) = \sum_{t=1}^{T} s[t]^2.$$

Calculate this energy for all training segments.

c. Wavelet band energies as features
We can also look at the energies of different frequency bands. Decompose each segment using the Daubechies wavelet of order four (`db4`) with five levels.

Calculate the energies contained in the approximation and detail coefficients for each segment. Check that you obtain an $N \times 6$ feature matrix: the energy of each of the six frequency bands, for all N segments.

d. Visualization of the features
Visualize your features using boxplots as in the previous session.

e. Classification of epileptic seizure activity
Calculate both the full energy of the test segments (as before) and the feature matrix of test segment subband energies (as before). Now solve two classification problems using the functions `fitcdiscr` and `predict`:

- Use the full signal energy as the only feature.
- Use the subband energies as features (collected in one feature vector).

f. Performance evaluation
Evaluate (accuracy, specificity, sensitivity) and compare both classifiers.

2.2.1 Additional questions

Answer, based on your plots and the tasks completed in this exercise, the following additional questions (number them in your report as they are listed here):

1. Given the plot(s) in 2.2a, what discriminant property do you observe? How would you quantify this?
2. Do you think we should normalize/standardize the raw data? Why (not)?
3. Given the wavelet decomposition in 2.2c, what are the frequency ranges corresponding to the approximation and detail coefficients? To what EEG rhythms do they correspond (approximately)?
4. Given the features in 2.2d, what frequency ranges/rhythms seem most interesting for classification?
5. Do you expect an improvement of the performance when adding the total signal energy (per segment) as an additional feature to the subband energies? Why (not)?

2.2.2 Deliverables

- Completed implementation of `ex5_2_2.m` and possibly additional code files.
- Discussion of all your results and plots.
- Plots of the overlaid seizure and non-seizure segments.
- Boxplots of different features.
- Overview and results of your classifiers.
- Check that you have answered all additional questions raised in Section 2.2.1 and make sure that they can be easily found back in your report.

References

[1] A. Hyvärinen and E. Oja, "Independent component analysis: A tutorial," 1999.

[2] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.