Exercise 3: Implementing CSP-based BCIs

Introduction to Modern Brain-Computer Interface Design

Christian A. Kothe
SCCN, UCSD

About

- This is a programming exercise in pure MATLAB (no toolboxes!)
- You will be implementing the critical parts of a CSP-based BCI: *the CSP calculation* and the *prediction function*
- The data is a recording of left/right hand imagined movements from the BCI competition 3

Preparation

- Start MATLAB
- To get into the directory, type:
  cd /your/path/to/exercise_package/ex3
- To open the evaluation script (that runs your code) type:
  edit run_evaluation

Details

- In this folder you find the scripts for this exercise
- **run_evaluation.m:** This function loads the data and runs your BCI functions
- **train_csp.m:** This is an incomplete scaffolding of the calibration function for this approach, here you need to fill in the code for the CSP calculation.
- **test_csp.m:** An incomplete scaffolding of the prediction function: you need to fill in the *complete prediction function.* Note: the role of the sliding data segment **X** is taken by the variable named **B**.
- ***Bonus:*** Once you get the correct mis-classification rate, extend the LDA calculation so that you get *gradual outputs* in approx. -1 to +1 range.

Exercise

- Implement the missing parts so that the run_evaluation function completes successfully
- The final output is the percentage of mis-classified trials in the data, it should be about 13% if your code is correct
- In the bonus part, your prediction function will then be applied in real time to played-back data – your prediction function will overshoot the -1 to +1 range by a wide margin if you do not do the Bonus exercise

Remarks

- The test_csp function works in a real online environment and can be fed with signal chunks of arbitrary length (i.e. whatever the EEG system delivers) – it internally aggregates the data in a sliding window **B** of fixed length (B is a global variable)

Tips

- **Tip 1:** if run_evaluation spends more than ca. 1 minute displaying “Predicting…”, you are probably applying the temporal filter to the 118-channel raw data instead of the 6-component spatially filtered data (which is *much* slower)
- **Tip 2:** You may want to take a look at the previous lecture slides to find the equations for what you will be implementing
- **Tip 3:** Be prepared to take a look at your intermediate results in the command line to fix any issues with wrong matrix shapes and so on
- Calculating the covariance matrix: cov()
- **Tip 4:** It is okay to calculate the covariance over the entire concatenated data for one class (instead of averaging cov’s for individual trials)